UNIVERSITÉ DE SHERBROOKE
Faculté des sciences appliquées
Département de génie chimique

CONTRÔLE ADAPTATIF PAR ENTRAÎNEMENT SPÉCIALISÉ DE RÉSEAUX DE NEURONES

Thèse de doctorat es sciences appliquées
Spécialité : génie chimique

Benoît Robitaille

Sherbrooke (Québec), CANADA Juin 1997

RÉSUMÉ

Cette thèse développe un algorithme de contrôle adaptatif constitué uniquement de réseaux de neurones. Cet algorithme utilise un entraînement spécialisé de réseaux de neurones. Par rapport aux autres approches qui utilisent un entraînement spécialisé, celle-ci se démarque par l'utilisation d'un réseau de neurones distinct pour modéliser la dynamique du procédé, par une utilisation cohérente des coordonnées de temps ainsi que par une nouvelle fonction erreur à minimiser qui est le reflet de la composante prédictive de l'algorithme. Cet algorithme est facile à utiliser, car il n'y a qu'un seul paramètre à ajuster pour le contrôleur.

Ce travail a permis de mettre au point un algorithme d'entraînement des réseaux de neurones répondant au besoin d'un contrôleur adaptatif qui suit un entraînement spécialisé. Cette méthode est une variante des méthodes quasi-Newton et prend en compte la structure du réseau dans la sélection des éléments d'interaction du deuxième ordre à évaluer. De plus, il n'y a aucun paramètre à ajuster pour faire l'entraînement.

Deux exemples d'utilisation du contrôleur neuronal adaptatif ont été développés. Les résultats montrent que le contrôleur a une bonne réponse aux changements de consigne et qu'il est capable de maintenir le procédé stable en présence de perturbations qui introduisent des écarts entre le modèle par réseau de neurones et le procédé. L'adaptation du contrôleur aux nouvelles conditions créées par les perturbations est rapide.

REMERCIEMENTS

En tout premier lieu, je tiens à remercier très sincèrement mon directeur de thèse, le professeur Bernard Marcos. Par l'intérêt qu'il a porté et le soutien qu'il m'a apporté tout au cours de ce projet de recherche, il a constamment démontré une rigueur et une analyse critique qui ont fait beaucoup plus qu'enrichir la thèse. Merci.

Je veux aussi remercier d'une façon spéciale les professeurs Guy Payre et Jean Lapointe qui ont démontré de l'intérêt pour les travaux que j'ai entrepris. Par les nombreuses discussions que nous avons eues, ils ont contribué de façon généreuse à la réalisation de cette thèse.

Je remercie les professeurs Pierre Proulx et Jules Thibault qui ont accepté très généreusement de consacrer du temps à l'examen de cette thèse.

Je me dois aussi de remercier les professeurs du département de génie chimique qui n'ont jamais hésité à partager leur temps et leurs expériences ainsi que tous les membres du personnel du département qui ont créé un cadre et une ambiance de travail si agréable.

Je m'en voudrais de ne pas remercier tous mes collègues du GRACO qui ont contribué grandement à la réalisation de ce travail et avec qui ce fut toujours un plaisir de collaborer ainsi que tous les membres des croûtons pour tous les moments si agréables que nous avons passés ensemble.

Enfin, je tiens à remercier mes parents et amis pour leur patience, leur compréhension et leur encouragement de tous les instants.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

1. INTRODUCTION

1.1 Les réseaux de neurones

L'utilisation de l'ordinateur est profondément ancrée dans les opérations de la vie courante en permettant d'effectuer des tâches longues ou répétitives ou bien des tâches complexes que nous ne saurions accomplir. Les ordinateurs sont rendus indispensables et nous ne pourrions imaginer évoluer sans eux. Nous devons toutefois constater que les ordinateurs ont encore beaucoup de difficultés à effectuer des tâches simples pour un humain telles que reconnaître et interpréter le sens des mots ou distinguer une forme d'une autre. Ce résultat peut sembler surprenant lorsque l'on sait que les neurones d'un cerveau sont très lents, comparés à la vitesse de calcul d'un ordinateur de l'ordre de la microseconde, un temps qui est considéré comme une éternité pour un microprocesseur moderne (Rich et coll. [1991]). Toutefois, comme le cerveau contient un très grand nombre de neurones et que ceux-ci travaillent en parallèle, il parvient à effectuer un raisonnement en très peu de temps.

Dans les années cinquante et soixante, ces observations ont initié les recherches portant sur les réseaux de neurones. Les travaux avaient pour objet d'utiliser le cerveau comme modèle afin de développer des architectures de traitement de l'information. Un modèle connexionniste très simple du fonctionnement du cerveau fut utilisé. Ce modèle est décrit ainsi : « *l'information que contient le cerveau est distribuée à travers un vaste réseau, interelié, d'éléments de traitements élémentaires appelés neurones* ». Bien que ce modèle simple reste valide, la science médicale d'aujourd'hui montre à quel point il n'était qu'une bien pâle image de la complexité et de la richesse du fonctionnement du cerveau.

Si les connaissances médicales ont évolué, les outils informatiques que sont les réseaux de neurones ont eux aussi beaucoup évolué depuis cette époque. Toutefois, une question demeure entière; est-il possible, en utilisant ce modèle, aussi simple soit-il, d'approcher les performances d'un être humain pour le traitement de l'information? Bien que peu de réalisations actuelles ne rivalisent en terme d'efficacité avec un cerveau humain, les percées effectuées dans des domaines

aussi variés que la reconnaissance de la parole (Lippmann [1989]), la classification de cellules cancéreuses (Moallemi [1991]), l'interprétation des biosenseurs (McAvoy et coll. [1989]), l'analyse des marchés boursiers (Tilford [1994]) et le pilotage automatique de camion (Nguyen et coll. [1990]) sont très encourageantes et justifient les efforts investis.

Le réseau de neurones, au sens informatique, est une structure parallèle distribuée de traitement d'informations. Il est constitué de plusieurs éléments de traitement, appelés neurones, interconnectés par des canaux unidirectionnels. Un neurone peut avoir plusieurs entrées mais n'a qu'une seule sortie. Cette sortie peut être à son tour déployée, à travers les canaux de connexion, vers plusieurs neurones. Toutes les informations nécessaires au calcul de la sortie d'un neurone doivent être contenues localement, c'est-à-dire l'évaluation de la valeur de sortie dépend uniquement des valeurs qui se présentent à l'entrée et de valeurs internes au neurone.

Afin d'avoir une certaine utilité, un réseau de neurones doit passer par une phase d'apprentissage durant laquelle les neurones s'adaptent pour reproduire les sorties désirées correspondant aux entrées présentées. Les mécanismes par lesquels est effectuée l'adaptation des neurones sont appelés les lois de l'apprentissage. Ces lois d'apprentissage sont regroupées en trois grandes classes générales: apprentissage supervisé, apprentissage par renforcement et apprentissage par organisation autonome. Lors de l'apprentissage supervisé, pour chaque entrée $\boldsymbol{X}$ présentée au réseau correspond une bonne valeur de sortie $\boldsymbol{Y}$. Dans l'apprentissage par renforcement, des entrées $\boldsymbol{X}$ sont présentées au réseau et à l'occasion, à intervalle régulier ou non, le réseau reçoit une indication de l'évolution de sa performance telle que mesurée depuis la dernière évaluation. Les réseaux de neurones qui font un apprentissage par organisation autonome reçoivent seulement des valeurs d'entrées et doivent s'organiser en une configuration utile, généralement par des mécanismes de compétition.

Parmi la très grande diversité de réseaux de neurones, ceux qui présentent un intérêt important pour le génie chimique sont regroupés sous le terme générique de réseaux de correspondance. Un réseau de neurones appartient à cette classe s'il est capable de calculer une relation fonctionnelle entre une entrée et une sortie. Un réseau de correspondance pourrait, par

exemple, établir la correspondance entre un nombre et la racine de ce nombre. Pour un exemple si simple, l'utilisation d'un réseau de neurones n'est nullement justifiée; par contre, pour une situation plus complexe, où la relation fonctionnelle n'est pas connue a priori mais dont des exemples de correspondance sont disponibles, les réseaux de neurones sont des outils de modélisation très puissants. Hecht-Nielsen [1990] explique que c'est l'habileté des réseaux de neurones à trouver leurs propres algorithmes de correspondance qui font leur force. C'est aussi la différence fondamentale entre les réseaux de neurones de correspondance et les méthodes de régression qu'elles soient linéaires ou non-linéaires. Dans tous les algorithmes de régression non linéaire, la première étape consiste à choisir un modèle de relation fonctionnelle, ou à tout le moins une famille de modèles, avant de procéder à l'étape de l'identification des paramètres.

1.2 Objectif des travaux de recherche

Depuis quelques années, le *Groupe de Recherche en Application Cognitive des Ordinateurs* (GRACO) de l'Université de Sherbrooke poursuit des recherches dans divers domaines de *l'intelligence artificielle* appliquée aux procédés. Des travaux ont été effectués dans les domaines du diagnostic de pannes par système expert ou par graphe orienté signé, d'autres dans le développement de tutoriels intelligents pour le transfert de connaissance dans des domaines technoscientifiques, et enfin dans le développement d'outils de caractérisation de l'eau et de modèle de dispersion de charge minérale dans des polymères extrudés à l'aide de réseaux de neurones.

L'objectif des travaux de recherche présentés dans cette thèse est de développer un contrôleur adaptatif constitué uniquement de réseaux de neurones. Pour atteindre cet objectif, un algorithme de contrôle qui utilise deux réseaux de neurones a été mis au point. Dans cet algorithme, le procédé est modélisé par un réseau de neurones qui suit un entraînement supervisé classique alors que la partie adaptative du contrôleur est constituée d'un réseau de neurones qui suit un entraînement supervisé spécialisé.

Comme tous les algorithmes de contrôle, le contrôle adaptatif par entraînement spécialisé de réseaux de neurones maintient le procédé de façon stable lors de perturbations et minimise les écarts à la consigne. De plus, puisqu'il est adaptatif, il est à même de suivre et de contrôler un procédé dont le comportement dérive dans le temps. Cet algorithme est facile à utiliser, car il n'y a qu'un seul paramètre à ajuster pour le contrôleur.

Pour entraîner le réseau de neurones qui suit un entraînement spécialisé, il faut une méthode d'entraînement qui soit performante, robuste et simple à implanter. De plus, comme l'entraînement se fait en continu, une méthode qui ne requiert pas d'ajuster les paramètres procure de la fiabilité au contrôleur. Parmi les techniques d'optimisation, et dans l'entraînement des réseaux de neurones en général, les méthodes quasi-Newton sont parmi les plus performantes et les plus fiables. Par contre, ces méthodes nécessitent d'emmagasiner beaucoup d'informations et leurs performances, lorsqu'elles sont utilisées dans un contexte d'entraînement de réseau de neurones, en sont souvent affectées. Afin de trouver un algorithme d'entraînement qui réponde au besoin du contrôleur adaptatif, une nouvelle méthode a été mise au point. Cette méthode est une variante des méthodes quasi-Newton. Elle tient compte de la structure du réseau dans la sélection des éléments d'interaction du deuxième ordre à évaluer, et n'a aucun paramètre à ajuster pour faire l'entraînement.

La thèse est divisée comme suit. Le chapitre 2 est consacré aux méthodes utilisées pour effectuer l'entraînement des réseaux de neurones. Plusieurs méthodes sont présentées, dont le très utilisé algorithme rétro-propagation, en mettant un effort particulier pour regrouper les méthodes à l'intérieur de trois grandes familles de méthodes d'optimisation: les méthodes gradient simple, les méthodes du gradient conjugué et les méthodes quasi-Newton.

Le chapitre 3 présente la nouvelle méthode d'entraînement. Trois configurations de cette méthode sont comparées aux méthodes rétro-propagation, gradient conjugué et méthode quasi-Newton avec mise à jour de la matrice hessienne BFGS au moyen de quatre exemples.

Le chapitre 4 présente, dans un premier temps, une revue des différentes approches utilisées pour trouver un modèle dynamique d'un procédé. Il expose ensuite une revue des techniques de contrôle de procédé en génie chimique qui utilisent des réseaux de neurones. Puis, il développe l'algorithme de contrôle adaptatif par entraînement spécialisé de réseaux de neurones, appelé contrôleur neuronal adaptatif.

Deux exemples d'utilisation du contrôleur neuronal adaptatif sont ensuite développés au chapitre 5. Il s'agit du contrôle d'un bioréacteur CFSTR, qui a une dynamique complexe avec différents états d'équilibre et un point de bifurcation. Dans l'autre exemple, un réacteur CSTR non isotherme avec une réaction réversible du premier ordre, sert de banc d'essai.

2. LES RÉSEAUX DE NEURONES À PROPAGATION AVANT

2.1 Introduction

Les réseaux de neurones à propagation avant trouvent plusieurs applications dans les domaines du génie chimique. Cette section présente en premier lieu le formalisme de la représentation des réseaux de neurones à propagation avant. Ensuite, des exemples d'utilisation de ces réseaux sont présentés. La partie suivante présente la règle du delta généralisé et l'évaluation du gradient nécessaire à l'entraînement des réseaux. Par la suite, plusieurs méthodes d'entraînement de réseaux de neurones qui ont été proposées dans les dix dernières années sont présentées. Les méthodes sont regroupées dans l'une des trois grandes familles qui rassemblent la vaste majorité des méthodes d'entraînement issues des techniques classiques d'optimisation. Ces trois familles sont : les méthodes du gradient simple, qui incluent l'algorithme rétro-propagation, les méthodes du gradient conjugué et les méthodes quasi-Newton.

2.2 Les réseaux de neurones à propagation avant, multicouche, entièrement connectés

Les réseaux de neurones qui établissent des correspondances servent donc à approximer des fonctions mathématiques. Les fonctions approximées peuvent être soit discrètes, comme dans les problèmes de classification, ou soit continues, comme dans l'approximation de fonctions phénoménologiques ou dans la corrélation de données expérimentales. Dans la majorité des cas, l'approximation obtenue sera de forme continue. Les réseaux de neurones de correspondance peuvent prendre une multitude de formes et d'aspects, et il existe des exemples de ces réseaux appartenant à chacune des trois grandes classes d'apprentissage (Hecht-Nielsen [1990]). Nous nous limiterons dans cette étude à une sous-classe des réseaux de neurones nécessitant un apprentissage supervisé : les réseaux de neurones à propagation avant, multicouches, et dont les neurones entre deux couches adjecentes sont entièrement connectés.

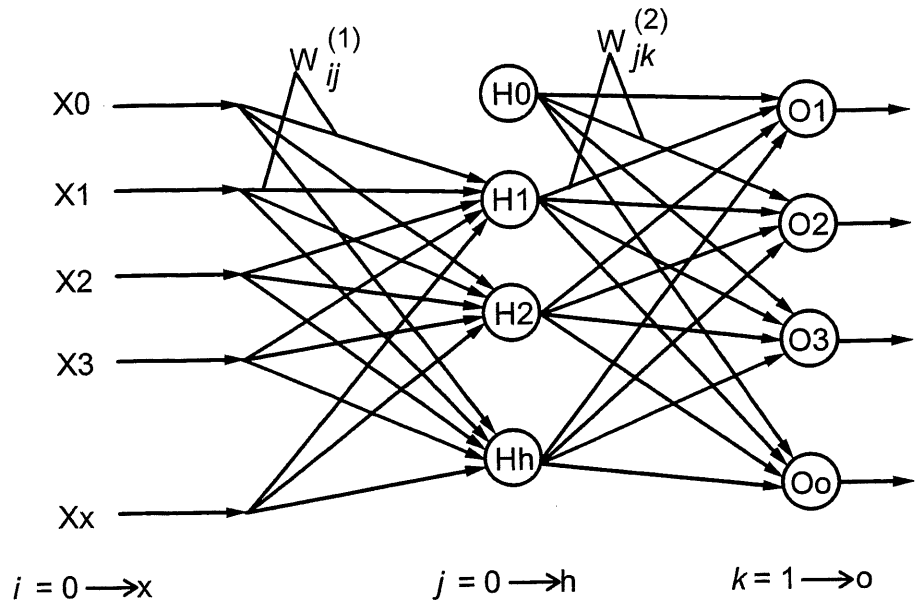


Figure 2.1 Schéma d'un réseau de neurones à propagation avant, multicouche et entièrement connecté

La figure 2.1 montre un exemple d'un réseau à trois couches qui établit la correspondance de $\mathbf{R}^x \rightarrow \mathbf{R}^o$. Les vecteurs [$X$, H, O] représentent respectivement les neurones d'entrée, de la couche cachée et de la sortie. Les poids associés aux branchements entre les niveaux sont $W^{(L)}$ pour le niveau L tandis que la fonction d'activation pour ce même niveau est $f^{(L)}$. Pour un réseau à une seule couche cachée, L prend la valeur de (1) ou de (2). Les valeurs des poids du réseau sont déterminées à partir d'un ensemble de P exemples d'entraînement (X_1 , Y_1) , (X_2 , Y_2) , ... , (X_P , Y_P) , où $Y_P = \phi(X_P)$ et ϕ est la fonction mathématique à approximer. X_0 et H_0 ont une valeur fixe, en général égale à un, et une fois multipliés par leurs poids associés, ils constituent le seuil d'activation des neurones du niveau suivant. Ces seuils d'activation, qui sont ajoutés lors du calcul des fonctions d'activation, sont optionnels. La pratique a démontré que leur présence ajoute généralement des degrés de liberté au système et accélère la convergence du procédé itératif nécessaire pour le calcul des poids $W^{(L)}$ du réseau.

Les fonctions d'activation sont de la forme suivante:

$$H_{pj} = f^{(1)}\left(\sum_{i=0}^{x} W_{ij}^{(1)} X_{pi}\right) \tag{2-1}$$

$$O_{pk} = f^{(2)}\left(\sum_{j=0}^{h} W_{jk}^{(2)} H_{pj}\right) \tag{2-2}$$

Le calcul des valeurs des poids $W^{(L)}$ permettant d'approximer la fonction ϕ est appelé l'apprentissage du réseau de neurones. Le nombre de poids que contient un réseau de neurones est exprimé par la variable *w*, et se calcule, lorsque des seuils d'activation sont utilisés, par:

$$w = (x+1).h \; + \; (h+1).o \tag{2-3}$$

La technique d'apprentissage pour un réseau de neurones multicouche à propagation avant requiert la présentation au réseau d'une paire de vecteurs entrée-sortie ($\boldsymbol{X_P}$, $\boldsymbol{Y_P}$). Le système utilise d'abord le vecteur d'entrée pour produire son propre vecteur de sortie $\boldsymbol{O_P}$ et le compare ensuite au vecteur de sortie $\boldsymbol{Y_P}$. On appelle une époque la présentation au réseau des *P* exemples d'entraînement. Les politiques suivies pour ajuster les poids constituent les règles d'apprentissage. Ces politiques sont en fait des techniques d'optimisation qui minimisent une fonction erreur des vecteurs $\boldsymbol{O}$ et $\boldsymbol{Y}$.

$$Ep = F(O_p, Y_p) \tag{2-4}$$

$$E = \sum_{p=1}^{P} Ep \tag{2-5}$$

Dans l'équation précédente, *Ep* est la fonction erreur à minimiser et la fonction *F* est habituellement la norme euclidienne. Cette minimisation requiert le calcul du gradient de la norme euclidienne du vecteur (*O* - *Y*) par rapport aux poids $W^{(L)}$ du réseau. Plusieurs auteurs auraient développé de façon indépendante la technique qui permet l'évaluation de ce gradient. Werbos [1974] fut le premier puis Parker [1985] et Rumelhart et coll. [1986] ont suivi. Le crédit

d'avoir développé un formalisme très simple, qui a popularisé la technique, revient sans contredit à Rumelhart et à son équipe du PDP « Parallel Distributed Processing ». Ils ont appelé règle delta généralisé la technique pour évaluer le gradient. Cette règle est présentée à la section 2.4 .

2.3 Exemple d'utilisation de réseaux de neurones

Les réseaux de neurones à propagation avant (Figure 2.1) en raison de leurs propriétés mathématiques, sont tout indiqués comme outils de modélisation. Des résultats mathématiques prouvent que ces réseaux de neurones sont des approximateurs généraux de fonctions et qu'une structure avec une seule couche cachée sera toujours suffisante pour représenter toute fonction arbitraire continue (Hecht-Nielsen [1990]). Par contre, ces résultats n'indiquent pas le nombre de neurones dans la couche cachée qui seront nécessaires pour réussir cette tâche.

Il existe dans la littérature une grande quantité d'exemples de modélisation qui utilisent des réseaux de neurones appliqués au génie chimique et, afin de démontrer l'étendue du domaine possible d'application, quelques-uns sont présentés.

Thibault et coll. [1991] ont utilisé un réseau de neurones pour modéliser une série de corrélations qui relient le nombre de Nusselt au nombre de Rayleigh lors de la convection naturelle le long d'un cylindre horizontal. Normandin et coll. [1993] ont employé un ensemble de deux réseaux de neurones pour corréler une quantité impressionnante de données : 1539 données de pression, de volume et de température de gaz et de vapeur. Les réseaux ont pour entrée la température réduite, la pression réduite et un facteur acentrique qui caractérisent les différents composés et pour sortie, une fonction du facteur de compressibilité. Deux réseaux ont été utilisés afin de bien modéliser chacune des régions importantes. Un autre domaine d'utilisation des réseaux de neurones qui gagne rapidement en popularité est celui des estimateurs d'états qui sont souvent appelés « logiciel senseur ». Ruenglertpanyakul et coll. [1992] ont développé un estimateur d'états à l'aide d'un réseau de neurones qui estime la concentration de cellule ainsi que la concentration de phénylalanine dans un bioréacteur « fed-batch ». Dans une application industrielle, Chitra [1993] a utilisé un réseau de neurones pour modéliser les effets du

chargement en catalyseur d'un réacteur batch par rapport aux constantes des taux de cinétique dans un procédé d'hydrogénation d'hydrocarbones.

2.3.1 La modélisation de la dynamique des procédés

La modélisation de procédés ayant une évolution dynamique occupe une place importante parmi les exemples d'utilisation de réseaux de neurones, particulièrement en génie chimique, en raison de l'importance traditionnelle qui est accordée à la modélisation pour la régularisation et le contrôle des procédés. Bhat et coll. [1990] ont modélisé le comportement dynamique du pH dans un réacteur CSTR et ont comparé leurs résultats avec une approche traditionnelle ARMA (auto-regressive moving average). Cette comparaison a montré que le réseau de neurones est capable de mieux caractériser le comportement nonlinéaire du système que la méthode ARMA. Dans un effort pour établir des parallèles entre les réseaux de neurones utilisés à des fins de contrôle et la théorie du contrôle adaptatif, Ydstie [1990] a modélisé un réacteur CSTR, dans lequel se produit une réaction du second ordre, et a utilisé le modèle obtenu dans un schéma de contrôle adaptatif à un pas en avant.

2.3.2 La classification par des réseaux de neurones

Plusieurs articles portant sur l'utilisation des réseaux de neurones pour la classification ont été publiés. La classification est l'exercice par lequel sont séparés des événements discrets. Citons, dans le domaine du diagnostic de pannes de procédés chimiques, Hoskin et coll. [1988], Ungar et coll. [1990], Venkatasubramania et coll. [1989] et [1990]. Ces travaux ont démontré la possibilité d'utiliser les réseaux de neurones pour classifier des événements. Malgré tout l'engouement existant autour de la technique, les travaux de Kramer et coll. [1991] n'ont pu démontrer d'avantages marqués à utiliser des réseaux de neurones lors d'exercice de classification d'événements discrets. Ils ont montré que d'autres méthodes, souvent beaucoup plus simples, donnent des résultats comparables ou même supérieurs en terme de pouvoir de séparation. Notamment la méthode qui consiste à associer un événement à la classe dont k événements se trouvent à la distance euclidienne la plus proche (souvent k=1) (« k nearest

neighbors ») donne d'excellents résultats. Ils constatent dans leur analyse que les travaux cités précédemment ont été réalisés avec des cas idéaux où les points d'entraînement couvrent l'ensemble du domaine dans lequel les points nouveaux peuvent se trouver. C'est aussi avec ce type de résultats que sont associés les concepts de généralisation souvent liés aux réseaux de neurones. Des difficultés surviennent lorsque de nouveaux cas se trouvent à l'extérieur du domaine d'entraînement et que le réseau de neurones doit extrapoler. Ces difficultés proviennent du fait que les réseaux de neurones associent arbitrairement des régions de classification aux espaces qui ne sont pas couverts par les points d'entraînement, et que les classes attribuées à ces espaces sont dépendantes du minimum local où se trouve le réseau à la fin de la période d'apprentissage. Kramer et coll. [1990] identifient cinq situations fréquentes où les réseaux de neurones ont de la difficulté à bien extrapoler à partir des points d'entraînement.

- Le nombre de points d'entraînement est trop restreint. Cela peut se produire pour plusieurs raisons: le manque de données expérimentales, le peu de points disponibles pour les régions à faible probabilité ou une capacité de calcul insuffisante.
- Un déplacement dans la distribution des classes qui se produit après l'entraînement; cela est toujours possible lorsque le système modélisé n'est pas parfaitement statique.
- Une panne de capteur corrompt les données d'entrées. Venkatasubramania et coll. [1989], préconisent de prendre une valeur par défaut lors d'une panne de capteur. Cette approche est très risquée car elle peut induire des erreurs subséquentes très difficiles à déceler.
- Un exemple d'une nouvelle classe apparaît après l'entraînement.
- Le réseau est entraîné à partir de données simulées plutôt que de données provenant d'expériences. Évidemment, tout dépend de la validité du modèle.

2.3.3 Le contrôle de procédé à l'aide de réseaux de neurones

Comme les réseaux de neurones sont de bons outils pour modéliser les procédés qui ont des dynamiques complexes à partir d'observations sur l'état des entrées et des sorties, ils sont tout à fait indiqués comme modèle dans plusieurs algorithmes de contrôle, comme le contrôle par modèle interne (Nahas et coll. [1992]) ou le contrôle adaptatif (Ydstie [1990]). Des réseaux de neurones ont été utilisés pour contrôler des procédés hautement non linéaires comme la régulation

du pH (Saint-Donat et coll. [1991]) et plusieurs procédés biochimiques (Ungar [1990]). Psichogios et coll. [1991] ont comparé l'utilisation de plusieurs algorithmes qui utilisent des réseaux de neurones comme modèles du procédé pour le contrôle d'un réacteur CSTR.

L'une des propriétés les plus intéressantes des réseaux de neurones est, qu'à partir des mêmes observations sur les entrées et sorties d'un procédé, on peut tout aussi bien établir le modèle de la dynamique directe que la dynamique inverse.

En plus de l'utilisation comme modèle, les réseaux de neurones trouvent aussi des applications comme contrôleurs avec ou sans possibilité d'adaptation. En fait, un modèle dynamique inverse s'apparente beaucoup à un contrôleur à anticipation où la prochaine commande est anticipée à partir de l'état du système. Pour les procédés dont la dynamique évolue dans le temps, le contrôleur ou le modèle doit s'adapter pour suivre l'évolution. Lorsqu'un réseau de neurones est utilisé comme contrôleur, cela implique une phase de réentraînement qui est soit périodique, comme dans les travaux de Ydstie [1990], soit en continu, comme dans les travaux de Psaltis et coll. [1987]. Un entraînement en continu d'un contrôleur est appelé un entraînement spécialisé puisque le contrôleur se *spécialise* dans la zone d'opération d'intérêt.

2.4 La règle delta généralisé

La règle du delta généralisé permet de calculer le gradient de la fonction erreur évaluée à la sortie d'un réseau de neurones par rapport aux poids entre les neurones du réseau. L'évaluation du gradient est l'étape préliminaire à tout algorithme d'entraînement des réseaux de neurones.

Soit U, un neurone à une position quelconque parmi les niveaux $[\boldsymbol{X},\boldsymbol{H},\boldsymbol{O}]$ de la figure 2.1, et soit i,j,k, les éléments d'un niveau tel que i est le niveau précédent de j et k est le niveau suivant. (Ici, les indices ne sont pas strictement associés à la couche d'entrée, aux couches cachées et la couche de sortie).

Soit E la fonction erreur à minimiser, qui est reliée à la norme euclidienne de la différence entre le vecteur d'entraînement $\boldsymbol{Y}$ et le vecteur de sortie du réseau, $\boldsymbol{O}$:

$$E = \sum_{p=1}^{P} Ep \tag{2-6}$$

pour une présentation p, telle que :

$$Ep = \frac{1}{2}\sum_{j}(Y_{pj} - O_{pj})^2 \tag{2-7}$$

Définissons la sortie d'un neurone :

$$U_{pj} = f_j(sum_{pj}) \tag{2-8}$$

évaluée par la fonction d'activation f, où :

$$sum_{pj} = \sum_{i} W_{ij} U_{pi} \tag{2-9}$$

Pour minimiser la fonction objective Ep, on pose que la variation des poids du réseau est proportionnelle à la direction opposée du gradient de la fonction objective par rapport aux poids du réseau.

$$\Delta_p W_{ij} \propto -\frac{\partial Ep}{\partial W_{ij}} \tag{2-10}$$

Cette dérivée est le résultat du produit de deux termes, tels que :

$$\frac{\partial Ep}{\partial W_{ij}} = \frac{\partial Ep}{\partial sum_{pj}} \frac{\partial sum_{pj}}{\partial W_{ij}} \tag{2-11}$$

Par l'équation (2-9), le second terme de cette équation est égal à :

$$\frac{\partial sum_{pj}}{\partial W_{ij}} = \frac{\partial \sum_{l} (W_{lj} U_{pl})}{\partial W_{ij}} = U_{pi} \tag{2-12}$$

en remarquant que

$$\frac{\partial \sum_{l} (W_{lj} U_{pl})}{\partial W_{ij}} = 0 \tag{2-13}$$

lorsque $l \neq i$ et où l représente le même niveau que i.

Par définition, posons :

$$\delta_{pj} = -\frac{\partial Ep}{\partial sum_{pj}} \tag{2-14}$$

l'équation (2-11) devient :

$$-\frac{\partial Ep}{\partial W_{ij}} = \delta_{pj} U_{pi} \tag{2-15}$$

Donc, pour minimiser l'erreur E selon le gradient, les poids doivent varier selon:

$$\Delta W_{ij} = \varepsilon \sum_{p=1}^{P} \delta_{pj} U_{pi} \tag{2-16}$$

où ε représente un pas de descente infinitésimal.

On doit maintenant déterminer la valeur de δ_{pj} pour chacun des neurones U_j du réseau. Les δ_{pj} sont fonctions du niveau auquel ils sont associés et de la fonction d'activation f_j .

En décomposant l'équation (2-14) par la règle d'enchaînement, on obtient :

$$\delta_{pj} = -\frac{\partial Ep}{\partial sum_{pj}} = -\frac{\partial Ep}{\partial U_{pj}} \frac{\partial U_{pj}}{\partial sum_{pj}} \tag{2-17}$$

Par l'équation (2-8), le second terme devient:

$$\frac{\partial U_{pj}}{\partial sum_{pj}} = f_j'(sum_{pj}) \tag{2-18}$$

qui est simplement la dérivée de la fonction d'activation de l'unité j évaluée à sum_{pj}. La valeur de cette dérivée sera présentée à la section 2.4.1 pour quelques fonctions d'activation.

Le premier terme est fonction du niveau auquel il est associé, et on obtient donc deux définitions, une pour la couche finale et une pour les couches intermédiaires. Pour la couche finale, on revient à la définition de *Ep*, soit l'équation (2-6):

$$Ep = \frac{1}{2}\sum_j (Y_{pj} - O_{pj})^2 \tag{2-19}$$

à partir de laquelle on calcule directement

$$\frac{\partial Ep}{\partial O_{pj}} = -(Y_{pj} - O_{pj}) \tag{2-20}$$

et le δ_{pj} devient

$$\delta_{pj} = f_j'(sum_{pj})(Y_{pj} - O_{pj}) \tag{2-21}$$

Pour la couche intermédiaire, on utilise la règle d'enchaînement pour écrire:

$$\frac{\partial Ep}{\partial H_{pj}} = \sum_k \frac{\partial Ep}{\partial sum_{pk}} \frac{\partial sum_{pk}}{\partial H_{pj}}$$
$$= \sum_k \frac{\partial Ep}{\partial sum_{pk}} \frac{\partial (\sum_l W_{lk} H_{pl})}{\partial H_{pj}} \qquad (2\text{-}22)$$
$$= \sum_k \frac{\partial Ep}{\partial sum_{pk}} W_{jk}$$

où l représente le même niveau que j

$$\frac{\partial Ep}{\partial H_{pj}} = -\sum_k \delta_{pk} W_{jk} \qquad (2\text{-}23)$$

et par l'équation (2-17)

$$\delta_{pj} = f_j'(sum_{pj}) \sum_k \delta_{pk} W_{jk} \qquad (2\text{-}24)$$

Le même développement s'applique quand on a plusieurs niveaux cachés pour écrire la variation de Ep en fonction d'un neurone intermédiaire comme une propagation de l'erreur de la couche précédente. Cette façon d'écrire les termes δ_{pj} implique que l'on doit les calculer couche par couche depuis la couche finale. La méthode propage l'erreur à rebours, ce qui explique le nom proposé de rétro-propagation par Rumelhart et coll. [1986].

En résumé, le gradient s'exprime pour la couche finale par:

$$\frac{\partial E}{\partial W_{ij}} = \sum_{p=1}^{P} -f'(sum_{pj}).(Y_{pj} - O_{pj}).U_{pi} \qquad (2\text{-}25)$$

et pour les couches intermédiaires par:

$$\frac{\partial E}{\partial W_{ij}} = \sum_{p=1}^{P} -f_j'(sum_{pj}).\sum_k(\delta_{pk}W_{jk}).U_{pi} \qquad (2\text{-}26)$$

2.4.1 Les fonctions d'activation

L'implantation de la règle delta généralisé nécessite que les fonctions d'activation soient croissantes ou décroissantes, bornées, continues et différentiables. Parmi les fonctions d'activation les plus utilisées se trouvent les fonctions sigmoïde et arctangente. La fonction sigmoïde est définie par:

$$f_j(sum_{pj}) = \frac{1}{1+e^{-sum_{pj}}} \qquad (2\text{-}27)$$

dont la dérivée s'écrit

$$f_j'(sum_{pj}) = f_j(sum_{pj}).(1 - f_j(sum_{pj})) \qquad (2\text{-}28)$$

et qui, pour la couche finale, est égale à:

$$f_j'(sum_{pj}) = O_{pj}.(1 - O_{pj}) \qquad (2\text{-}29)$$

La figure 2.2 présente l'allure générale de la fonction sigmoïde. Cette fonction est bornée par 0 et 1.

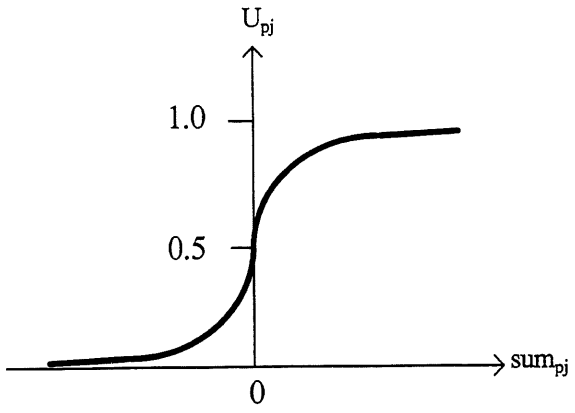


Figure 2.2 Fonction sigmoïde

La fonction arctangente est définie par:

$$f_j(sum_{pj}) = arct(sum_{pj}) \tag{2-30}$$

dont la dérivée est:

$$f_j'(sum_{pj}) = [1-(f_j(sum_{pj}))^2] \tag{2-31}$$

Cette fonction est bornée par -1 et 1 a un graph similaire à celui de la fonction sigmoïde tracée à la figure 2.2.

Toutes les fonctions d'activation bornées utilisées dans les réseaux de neurones ont une propriété similaire; lorsque la fonction d'activation approche une borne, c'est-à-dire qu'elle sature, la valeur de sa dérivée $f'(sum_{pj})$ tend vers zéro. Cela a pour effet de produire un gradient $\frac{\partial E}{\partial W_{ij}}$ dans les équations (2-25) et (2-26), qui tend vers zéro, même lorsque l'erreur à la couche finale est importante. Dans certains cas, et pour toutes les méthodes d'apprentissage faisant appel au calcul du gradient, cela ralentit considérablement la procédure d'optimisation. Ces cas se

produisent plus particulièrement lorsque le réseau est utilisé pour produire une sortie binaire et quand les valeurs des vecteurs d'entraînement $\boldsymbol{Y}$ sont près des bornes. On évitera donc toujours d'utiliser les seuils de saturation des fonctions d'activation comme valeurs cibles lors de l'entraînement.

2.5 Les techniques d'optimisation

Les méthodes présentées dans cette section sont développées pour des réseaux de neurones qui ont une seule couche cachée et qui suivent la notation de la figure 2.1. L'utilisation d'une seule couche cachée a pour but de simplifier la notation. L'expansion de tous les modèles pour les réseaux à couches cachées multiples est directe en utilisant les règles de rétropropagation qui ont été présentées lors du développement de la règle delta généralisé à la section 2.4 .

Pour cette section, i, j et k sont les éléments correspondant aux couches [$\boldsymbol{X}, \boldsymbol{H}, \boldsymbol{O}$] et i', j' sont les éléments de couches quelconques successives.

2.5.1 Évaluation du gradient

Les neurones H_{pj} et O_{pk} sont évalués par les équations (2-1) et (2-2) respectivement. Définissons

$$sum_{pj} = \sum_{i=0}^{a} W_{ij}^{(1)} X_{pi} \tag{2-32}$$

et

$$sum_{pk} = \sum_{j=0}^{b} W_{jk}^{(2)} H_{pj} \tag{2-33}$$

Le gradient s'exprime donc pour les poids de la couche finale comme:

$$\frac{\partial E}{\partial W_{jk}^{(2)}} = \sum_{p=1}^{P} -f'^{(2)}(sum_{pk}).(Y_{pk} - O_{pk}).H_{pj} \tag{2-34}$$

et pour les couches intermédiaires par:

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \sum_{p=1}^{P} -f'^{(1)}(sum_{pj}).\sum_{k=1}^{o}[f'^{(2)}(sum_{pk})(Y_{pk} - O_{pk})W_{jk}^{(2)}].X_{pi} \tag{2-35}$$

Réécrivons les gradients en utilisant une notation compacte et en ajoutant un indice d'itération (n). Le gradient associé à un poids et pour une présentation s'écrira ainsi :

$$Gp_{i'j'}^{(L)}(n) = \frac{\partial Ep}{\partial W_{i'j'}^{(L)}(n)} \tag{2-36}$$

et la somme des gradients pour l'ensemble des présentations par rapport à un même poids s'écrira:

$$G_{i'j'}^{(L)}(n) = \sum_{p=1}^{P} \frac{\partial Ep}{\partial W_{i'j'}^{(L)}(n)} \tag{2-37}$$

Définissons $\boldsymbol{G(n)}$ comme le vecteur gradient dont les composants $G_m(n)$ sont égaux à $G_{i'j'}^{(L)}(n)$ (Les caractères gras font référence aux vecteurs).

$$G_m(n) = G^{(L)}_{i'j'}(n) \tag{2-38}$$

$$\begin{aligned} m &= (j'-1)(x+1) + i' + 1 && 1 \le m \le (x+1)h \\ & && i' \in [0,x] \\ & && j' \in [1,h] \\ m &= (j'-1)(h+1) + i' + 1 + (x+1)h && (x+1)(h+1) \le m \le (x+1)h + (h+1)o \\ & && i' \in [0,h] \\ & && j' \in [1,o] \end{aligned}$$

2.5.2 Méthodes du gradient simple et l'algorithme rétro-propagation

Les méthodes présentées dans cette section ont toutes en commun un ajustement des poids seul à seul qui néglige complètement les interactions du second ordre entre les poids susceptibles d'influencer la fonction objective.

La méthode d'optimisation la plus simple à imaginer lorsque le gradient d'une fonction à minimiser est connu, est de déterminer une direction de descente $\mathbf{S}(n)$ opposée à la direction du gradient et d'effectuer une recherche unidirectionnelle dans cette direction. C'est la méthode d'optimisation du gradient simple qui s'écrit par:

$$\Delta \boldsymbol{W}(n) = \lambda(n)\boldsymbol{S}(n) \tag{2-39}$$

$$\boldsymbol{S}(n) = -\boldsymbol{G}(n) \tag{2-40}$$

et où $\Delta \boldsymbol{W}(n)$ est la variation des poids pour une itération, et $\lambda(n)$ est le coefficient trouvé à l'aide d'une technique de recherche unidirectionnelle qui minimise la fonction dans la direction de descente. Des essais ont démontré qu'il n'est pas nécessaire, ni même avantageux, de calculer $\lambda(n)$ de sorte que la fonction objective soit minimisée de façon exacte dans la direction de descente (Denis et coll. [1983]). Il est très coûteux de calculer le minimum exact selon une direction de descente, ce qui explique la popularité des algorithmes de recherche unidirectionnelle qui calculent $\lambda(n)$ en respectant des critères qui assurent une minimisation suffisante, tel le critère

d'Armijo (Armijo [1966]). L'algorithme de rétro-propagation développé par Rumelhart et coll. [1986] est basé sur une variation de la méthode du gradient simple.

$$\Delta_p W_{i'j'}^{(L)}(n) = -\eta G p_{i'j'}^{(L)}(n) + \alpha \Delta_p W_{i'j'}^{(L)}(n-1) \tag{2-41}$$

Cet algorithme n'effectue pas de recherche unidirectionnelle mais choisit pour λ une quantité constante η appelée la vitesse d'apprentissage ; il ajoute alors une fraction α de la variation de l'itération précédente. Cette seconde constante, appelée le momentum, sert à limiter la variation de la direction de descente d'une itération à l'autre. Cette façon de garder en mémoire l'information des itérations précédentes contient des éléments de la méthode du gradient conjugué. Dans l'équation (2-41), les poids sont ajustés à chaque présentation d'un couple de vecteurs d'entraînement $[\boldsymbol{X},\boldsymbol{Y}]$. Rumelhart et coll. [1986] appellent cette procédure l'ajustement continu. Ils ont aussi développé un algorithme qui effectue l'ajustement des poids après avoir présenté l'ensemble des vecteurs d'entraînement ; cet algorithme est nommé ajustement périodique et correspond à l'équation (2-42).

$$\Delta \boldsymbol{W}(n) = -\eta \boldsymbol{G}(n) + \alpha \Delta \boldsymbol{W}(n-1) \tag{2-42}$$

Il faut remarquer qu'après une époque, les deux formules représentées par les équations (2-41) et (2-42) ne donnent pas les mêmes résultats. Cela provient du fait que les deux formules ne minimisent pas les mêmes fonctions objectives. L'équation (2-41) minimise la fonction objective décrite par l'équation (2-5) alors que la formule (2-42) minimise la fonction objective décrite par l'équation (2-6).

Dans ces algorithmes, le fait que le pas de descente soit constant ne garantit pas que chaque itération minimise bel et bien la fonction objective, et un nombre important de calculs peut être effectué inutilement. De plus, il n'est aucunement garanti que la procédure va s'arrêter dans un minimum de la fonction objective, soit-il local ou global. Par contre, dans une situation où la topologie de la fonction objective est très particulière, l'algorithme peut escamoter des minimums locaux. Il s'agit en fait d'un faible avantage qui se produit de façon ponctuelle et qui

est pratiquement impossible à prévoir. L'avantage majeur de cette approche est que chaque itération est très rapide, puisqu'il n'y a pas de recherche unidirectionnelle comme avec la méthode du gradient simple, et que la méthode nécessite peu d'espace mémoire, soit deux vecteurs de dimension égale au nombre de poids totaux dans le réseau.

La principale difficulté avec ces algorithmes est de trouver la bonne valeur pour les constantes η et α. Selon la littérature, la valeur de η se situe entre 0.3 et 0.9 alors que les valeurs pour la constante de momentum vont de 0.0 à 0.9.

Une méthode qui a obtenu un certain succès, et qui est inspirée de l'algorithme de rétro-propagation, est l'algorithme « Quickprop » développé par Fahlman [1988].

$$\Delta W_{i'j'}^{(L)}(n) = -\varepsilon G_{i'j'}^{(L)}(n) + \frac{G_{i'j'}^{(L)}(n)}{G_{i'j'}^{(L)}(n-1) - G_{i'j'}^{(L)}(n)} \Delta W_{i'j'}^{(L)}(n-1) \qquad (2\text{-}43)$$

« Quickprop », tout comme rétro-propagation, ne fait pas de recherche unidirectionnelle mais utilise un pas de descente ε constant. Par contre, le terme de momentum est remplacé par une expression qui ajuste automatiquement sa valeur en fonction de celle de la courbure de la descente entre chaque itération et pour chaque poids. Cette méthode est en fait dérivée des méthodes bien connues qui sont basées sur l'approximation de Newton, mais elle néglige les interactions du deuxième ordre entre les poids

$$\Delta W_{i'j'}^{(L)} = -\frac{f'(E)}{f''(E)} \qquad (2\text{-}44)$$

pour lesquelles les dérivées secondes sont approximées par la différence des dérivées premières entre deux itérations. Cette approximation de la dérivée seconde conduit à la formulation de la méthode de la sécante:

$$\Delta W_{i'j'}^{(L)}(n) = \lambda \frac{G_{i'j'}^{(L)}(n)}{G_{i'j'}^{(L)}(n-1) - G_{i'j'}^{(L)}(n)} \Delta W_{i'j'}^{(L)}(n-1) \tag{2-45}$$

Fahlman [1988] combine donc, dans l'algorithme « Quickprop », la méthode de la sécante avec λ=1 et le gradient simple avec un pas de descente constant.

Pour les cas où les sorties sont binaires, Van Ooyen et coll. [1992] ont proposé de remplacer la fonction erreur traditionnelle, qui est la norme euclidienne, par une fonction qui a la forme:

$$E = -\sum_{p=1}^{P}\sum_{k=1}^{o}[O_j \ln(Y_k) + (1-O_k)\ln(1-Y_k)] \tag{2-46}$$

Cette fonction, lorsqu'elle est utilisée conjointement avec une fonction d'activation sigmoïde, conduit à un gradient pour la couche finale qui s'exprime par:

$$\frac{\partial E}{\partial W_{jk}} = \sum_{p=1}^{P}(Y_{pk} - O_{pk}).H_{pj} \tag{2-47}$$

qui est équivalent à l'équation (2-34) pour laquelle le terme de la dérivée de la fonction d'activation $f'^{(2)}(sum_{pk})$ a disparu. Comme il a été montré à la section 2.4.1, la dérivée d'une fonction d'activation tend vers zéro lorsque la fonction tend vers une borne. Cela a pour effet d'induire un gradient très faible, lorsque le gradient est évalué par la règle delta généralisé tel que décrit à la section 2.4, même s'il existe un écart important entre la valeur prédite par le réseau de neurones et la valeur désirée. Pendant la phase d'optimisation d'un réseau, si une valeur de sortie binaire tend vers le mauvais extrême, le faible gradient qui en résulte diminue considérablement l'importance à accorder à l'ajustement des poids face à cette valeur d'entraînement et prolonge donc le nombre d'itérations nécessaires pour obtenir une bonne minimisation du réseau. Le gradient évalué par l'équation (2-47) contourne cette difficulté en étant uniquement fonction, par rapport au neurone de la couche finale, à l'écart entre celle-ci et la valeur désirée. Les résultats

obtenus montrent une nette amélioration de la vitesse de convergence pour des exemples ayant une sortie binaire.

Jacobs [1988] a proposé d'utiliser des pas de descente indépendants associés à chaque poids. La valeur de ces pas est ajustée dynamiquement à chaque itération à partir d'heuristiques basées sur des informations comme l'évolution du signe d'un poids dans le temps. Tollenaere [1990] a présenté l'algorithme « SuperSAB » qui applique une stratégie d'adaptation dynamique à partir d'heuristiques basées sur les travaux de Jacobs[1988] mais en ajoutant un terme de momentum à l'algorithme de minimisation.

2.5.3 Méthode du gradient conjugué

La méthode du gradient conjugué fut d'abord proposée pour résoudre des systèmes d'équations linéaires et Fletcher et Reeves [1964] on adapté les concepts au cadre de l'optimisation non-linéaire. Une extension en fait l'une des méthodes d'optimisation sans contrainte les plus utilisée, plus spécialement pour les problèmes à grande échelle et les problèmes à matrice creuse. Cette méthode utilise des directions de descentes conjuguées évaluées à partir du gradient et du résidu. Deux directions, $\boldsymbol{S}^i$ et $\boldsymbol{S}^j$, sont dites conjuguées par rapport à la matrice H si :

$$[S^i]^T \mathbf{H} [S^j] = 0 \qquad (2\text{-}48)$$

Dans les problèmes d'optimisation, $\boldsymbol{H}$ est la matrice hessienne de la fonction objective.

Fletcher et coll. [1964] ont démontré que cette méthode a des propriétés de convergence quadratique lorsque qu'appliquée à des fonctions objectives quadratiques et lorsque la direction de descente est minimisée exactement à chaque itération. Elle offre une amélioration majeure par rapport aux méthodes du gradient simple tout en nécessitant seulement un léger effort de calcul supplémentaire. Par la méthode du gradient conjugué, les poids du réseau sont modifiés à chaque itération en effectuant une recherche unidirectionnelle dans la direction de descente $\boldsymbol{S}(n)$ de sorte que :

$$\Delta W(n) = \lambda(n)\, S(n) \tag{2-49}$$

La direction de descente $S(n)$ est déterminée par:

$$\mathbf{S}(n) = -\mathbf{G}(n) + \frac{\|\mathbf{G}(n)\|}{\|\mathbf{G}(n-1)\|}\mathbf{S}(n-1) \tag{2-50}$$

où $G(n)$ et $G(n\text{-}1)$ sont les gradients calculés à l'itération n et n-1.

Lorsque le pas de descente unidirectionnelle $\lambda(n)$ est égal à la vitesse d'apprentissage η et que le rapport des normes des gradients $\frac{\|\mathbf{G}(n)\|}{\|\mathbf{G}(n-1)\|}$ de deux itérations successives est égal au terme de momentum α, la méthode du gradient conjugué formulée par les équations (2-49) et (2-50) ressemble beaucoup à l'algorithme de rétro-propagation, équation (2-42).

Pour que les directions de descente demeurent vraiment conjuguées d'une itération à l'autre, la recherche unidirectionnelle doit être effectuée avec une très grande précision. Néanmoins, après plusieurs itérations, les directions de descente peuvent devenir pratiquement parallèles ce qui entraîne la perte de la propriété de convergence quadratique. Pour remédier à cette difficulté, Fletcher et coll. [1964] ont suggéré de repartir la procédure en remplaçant la direction de descente $S(n)$ par le gradient $G(n)$ à toute les $(t+1)$ itérations où (t) est le nombre total de variables à minimiser. Le choix de la précision de la recherche unidirectionnelle est un critère très sensible sur les performances de l'algorithme. Un critère très serré n'est pas nécessairement optimum, même s'il permet de réinitialiser la procédure beaucoup moins souvent. D'un autre côté, une recherche très imprécise peut engendrer une direction de descente erronée, c'est-à-dire une direction qui augmente la fonction objective.

Les méthodes s'inspirant du gradient conjugué sont très utilisées pour entraîner des réseaux de neurones. Citons, entre autres, les travaux de Nahas et coll. [1992] qui utilisent cette approche pour l'apprentissage de la dynamique de réacteurs chimiques. La méthode du gradient

conjugué décrite par Fletcher et coll. [1964] a été utilisée par Leonard et coll. [1990] pour entraîner des réseaux de neurones comme alternative à l'algorithme de rétro-propagation. Les méthodes ont été comparées sur deux exemples, le problème du XOR et le problème de diagnostic de pannes présenté par Hoskin et coll. [1988]. Curieusement, la comparaison en terme de temps de calcul nécessaire pour converger entre la méthode du gradient conjugué et l'algorithme de rétro-propagation n'est pas à l'avantage de la première, puisque les deux méthodes ont produit des résultats similaires pour le problème du XOR et que l'algorithme de rétro-propagation a été le plus performant pour le problème de diagnostic.

Kinsella [1992] a étudié une variante de l'algorithme du gradient conjugué qui évite la recherche unidirectionnelle. La technique consiste à déterminer les directions conjuguées à partir d'une approximation de la matrice hessienne que l'on forcera à demeurer définie positive par une approche de type Levenberg-Marquardt. Lorsque la matrice hessienne est définie positive, on peut localement remplacer la fonction à minimiser par une fonction quadratique. On trouvera alors par un calcul direct le scalaire qui permet de minimiser cette fonction quadratique dans la direction de conjuguée à la direction d'arrivée par rapport à la matrice hessienne. Malheureusement, les résultats présentés avec cette approche sont moins intéressants que ceux obtenus par l'algorithme traditionnel du gradient conjugué.

2.5.4 Méthodes quasi-Newton

Lorsque l'on désire minimiser une fonction plus on a une idée précise de sa topologie, plus on peut choisir une direction de descente efficace. En plus du calcul du gradient, l'information la plus utile pour évaluer la topologie d'une fonction est la dérivée seconde de la fonction objective par rapport à chacun des paramètres. Définissons d'abord la variation des poids d'un réseau de neurones par rapport à une direction de descente :

$$\Delta \boldsymbol{W}(n) = \lambda(n)\, \boldsymbol{S}(n) \tag{2-51}$$

où $\lambda(n)$ est évalué par une recherche unidirectionnelle et où la direction de descente est déterminée par:

$$\boldsymbol{S}(n) = -[\boldsymbol{H}(n)]^{-1} \quad \boldsymbol{G}(n) \tag{2-52}$$

Bishop [1992] a publié une procédure itérative qui permet le calcul exact de la matrice hessienne. Bien que cette procédure s'avère utile pour l'évaluation des plages d'erreurs à la sortie d'un réseau ou pour toute autre évaluation statistique du comportement d'un réseau, elle est très coûteuse en temps de calcul et, par conséquent, difficilement utilisable dans un algorithme d'apprentissage d'un réseau de neurones.

Le calcul de la direction de descente donné par l'équation (2-52) nécessite l'inversion de la matrice hessienne $\boldsymbol{H}$(n) une procédure qui est très fastidieuse à réaliser à chaque itération de la procédure de minimisation. Les méthodes quasi-Newton permettent de contourner cette difficulté en approximant directement la matrice hessienne inverse $[\tilde{\boldsymbol{H}}(\mathrm{n})]^{-1}$ à partir des dérivées premières de la fonction objective, $\boldsymbol{G}(n)$. Denis et coll. [1983] ont proposé plusieurs formulations et ont présenté une très bonne démonstration de la théorie qui a mené aux développements des méthodes quasi-Newton. Ces méthodes sont sans contredit les plus populaires en optimisation sans contraintes. Plusieurs formules d'approximation ont été développées et les plus siuvent mises en œuvre sont BFGS (Broyden, Fletcher, Goldfarb, Shanno) et dans un ordre moindre, DFP (Davidson, Fletcher, Powell). Seule la formule BFGS sera présentée ici.

Himmelblau [1990] a montré comment écrire la matrice hessienne inverse approximée, $[\tilde{\boldsymbol{H}}(\mathrm{n})]^{-1}$, par la méthode BFGS dans le cas d'un réseau de neurones.

$$\Delta[\tilde{\boldsymbol{H}}(n)]^{-1} = [\tilde{\boldsymbol{H}}(n+1)]^{-1} - [\tilde{\boldsymbol{H}}(n)]^{-1} \tag{2-53}$$

$$\Delta[\tilde{\boldsymbol{H}}(n)]^{-1} = \frac{[\Delta \boldsymbol{W}(n) - [\tilde{\boldsymbol{H}}(n)]^{-1}\Delta \boldsymbol{G}(n)][\Delta \boldsymbol{W}(n)]^T + [\Delta \boldsymbol{W}(n)][\Delta \boldsymbol{W}(n) - [\tilde{\boldsymbol{H}}(n)]^{-1}\Delta \boldsymbol{G}(n)]^T}{[\Delta \boldsymbol{G}(n)]^T \Delta \boldsymbol{W}(n)} - \frac{[\Delta \boldsymbol{W}(n) - [\tilde{\boldsymbol{H}}(n)]^{-1}\Delta \boldsymbol{G}(n)]^T \Delta \boldsymbol{G}(n)\, \Delta \boldsymbol{W}(n)\, [\Delta \boldsymbol{W}(n)]^T}{[[\Delta \boldsymbol{G}(n)]^T \Delta \boldsymbol{W}(n)][[\Delta \boldsymbol{G}(n)]^T \Delta \boldsymbol{W}(n)]} \tag{2-54}$$

Watrous [1987] a comparé les méthodes BFGS et DFP à celles de rétro-propagation et du gradient simple avec recherche unidirectionnelle pour l'entraînement de réseaux de neurones. Il a démontré, dans deux exemples, que le nombre d'évaluations de la fonction et du gradient pour minimiser la norme euclidienne est nettement plus faible avec la méthode BFGS. L'inconvénient des méthodes quasi-Newton consiste à devoir emmagasiner de l'ordre de $(\boldsymbol{W})^2$ valeurs à chaque itération par rapport à $(\boldsymbol{W})$ pour les méthodes de gradient simple ou de gradient conjugué ; chaque itération des méthodes de quasi-Newton nécessite plus de calcul que les méthodes de gradients. Lorsque la taille des réseaux est grande, Watrous [1987] et Nahas et coll. [1992] considèrent que cet inconvénient n'incite pas à utiliser des méthodes quasi-Newton et que ces méthodes sont limitées aux réseaux de taille moyenne. Pour remédier à cet inconvénient, Becker et coll. [1988] ont proposé une approximation très simple de la matrice hessienne qui ne considère que les termes situés sur la diagonale. Cette approximation n'a pas réduit de façon significative le temps nécessaire pour optimiser les poids d'un réseau. Kollias et coll. [1989] ont modifié la méthode de Levenberg-Marquardt pour utiliser une approximation de la matrice hessienne constituée des termes situés près de la diagonale. Cette approximation conduit à solutionner à chaque itération un système linéaire qui peut être résolu de façon parallèle. Toutefois, lors de calculs en série, la méthode apporte peu d'avantage en terme de temps de calcul par rapport à l'algorithme de rétro-propagation.

3. UNE MÉTHODE QUASI-NEWTON MODIFIÉE POUR L'APPRENTISSAGE DES RÉSEAUX DE NEURONES

3.1 Introduction

Plusieurs auteurs ont déclaré que les méthodes quasi-Newton sont limitées aux réseaux de neurones de dimension moyenne à cause de l'espace mémoire et du temps de calcul nécessaires pour effectuer la mise à jour de l'approximation de la matrice hessienne (Watrous [1987], Nahas et coll. [1992]). Ce chapitre présente des modifications à l'approche classique des méthodes quasi-Newton, modification ayant pour but d'accélérer le temps nécessaire à l'entraînement d'un réseau de neurones et de réduire l'espace mémoire requis pour emmagasiner l'information à chaque itération. Ces modifications consistent en de nouvelles approximations de la matrice hessienne qui négligent certaines interactions du deuxième ordre et qui sont construites en tenant compte de la structure du réseau de neurones. De ces modifications, trois méthodes pour l'optimisation des poids d'un réseau de neurones sont développées. Elles sont ensuite comparées à d'autres méthodes traditionnelles tels l'algorithme de rétro-propagation, le gradient conjugué et la méthode quasi-Newton BFGS.

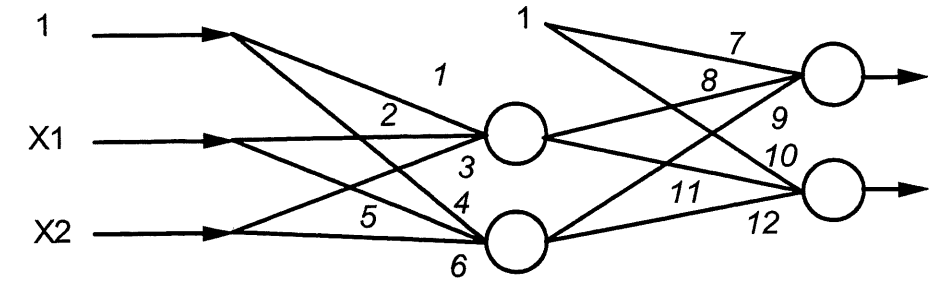
3.2 Une formulation simplifiée de la matrice hessienne

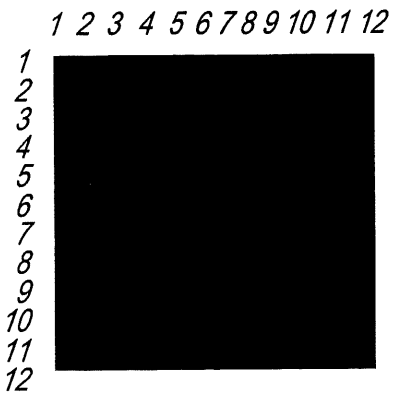
Les trois méthodes d'optimisation utilisées pour l'apprentissage de réseaux de neurones sont développées à partir d'une simplification de termes dans l'approximation de la matrice hessienne, chacune d'elles négligeant un nombre différent d'interactions en fonction de la structure du réseau de neurones. Négliger des interactions du deuxième ordre dans la méthode quasi-Newton se traduit par une direction de descente qui est moins efficace et donc par une augmentation du nombre d'itérations nécessaires pour obtenir une solution. Cet effet va être compensé par une réduction dans le temps de calcul demandé pour renouveler l'approximation de la matrice hessienne et par une diminution de l'espace mémoire requis. En étudiant trois différentes configurations, nous recherchons le meilleur compromis entre la dimension de la matrice hessienne approximée et le temps de calcul nécessaire pour optimiser les poids d'un

réseau de neurones. Les mises à jour des approximations des matrices hessiennes se font par la méthode BFGS.

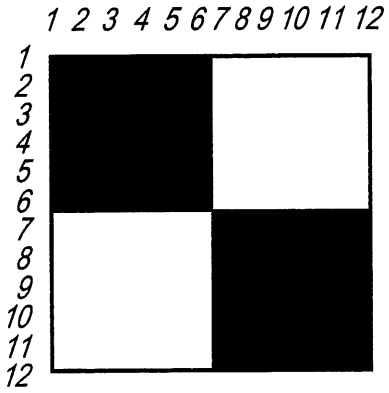
La première méthode néglige les interactions du second ordre entre les poids des différents niveaux d'un réseau de neurones et considère des matrices $\tilde{H}$ indépendantes associées à chaque niveau. Cette méthode se nomme BFGS-L. Les travaux de Bishop [1992] ont montré qu'il n'y a pas d'interaction entre les neurones de sortie d'un réseau, ce qui veut dire que la matrice hessienne exacte est « creuse » par rapport aux poids de la couche finale. La seconde méthode s'inspire de ces résultats et modifie la méthode BFGS-L en attribuant une matrice $\tilde{H}$ pour chaque neurone du niveau de sortie tout en conservant une matrice $\tilde{H}$ pour chaque niveau intermédiaire. Cette méthode se nomme BFGS-M. La troisième méthode, nommée BFGS-N, considère uniquement les interactions du deuxième ordre entre les poids associés à un même neurone et évalue donc des matrices $\tilde{H}$ distinctes pour chaque neurone du réseau. Pour un réseau de neurones qui a un seul neurone de sortie, les méthodes BFGS-L et BFGS-M sont identiques.

Le réseau de neurones de la figure 3.1 a deux entrées, deux neurones dans la couche cachée et deux neurones de sortie, ce qui fait un total de 12 poids en tenant compte des poids associés au seuil d'activation. Pour ce réseau, la matrice $\tilde{H}$ provenant de la méthode BFGS contient $(12)^2 = 144$ éléments ; la méthode BFGS-L en contient $(6)^2 + (6)^2 = 72$; la méthode BFGS-M en contient $(6)^2 + (3)^2 + (3)^2 = 54$; la méthode BFGS-N en contient $(3)^2 + (3)^2 + (3)^2 + (3)^2 = 36$. Par exemple, la dérivée seconde , $\frac{\partial^2 E}{\partial W_5 \partial W_9}$, n'a pas à être évaluée pour les méthodes BFGS-L, BFGS-M et BFGS-N. Les matrices hessiennes approximées par les méthodes quasi-Newton sont toujours symétriques et donc seule la partie triangulaire supérieure doit être calculée. L'avantage d'associer des matrices $\tilde{H}$ avec chaque niveau ou chaque neurone est la réduction considérable de la matrice hessienne approximée résultante qui doit être évaluée. La figure 3.1 montre la dimension des matrices hessiennes provenant d'un réseau de neurones pour les configurations suivantes : BFGS, BFGS-L, BFGS-M et BFGS-N.

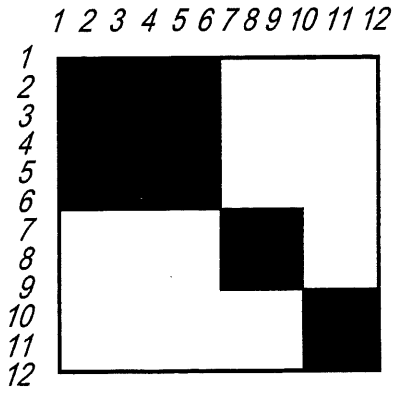




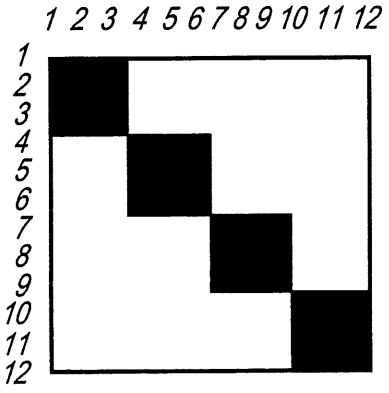
BFGS



BFGS-L



BFGS-M



BFGS-N

Figure 3.1 Structure des matrices hessiennes appliquées à un réseau de neurones pour quatre méthodes d'approximation.

3.2.1 Formulation du vecteur de recherche

Dans ces 3 méthodes, le calcul de la variation des poids est similaire à celui utilisé dans les méthodes quasi-Newton. Il diffère seulement dans les éléments choisis pour former le vecteur gradient utilisé dans l'évaluation de l'approximation de la matrice hessienne. Pour la méthode BFGS-L qui considère uniquement les interactions du deuxième ordre entre les poids d'un même niveau, on définit un vecteur gradient $\boldsymbol{G}^{(L)}$ qui a pour éléments tous les $\boldsymbol{G}_{i'j'}^{(L)}$ d'un même niveau et de là, on peut construire le gradient global du système comme étant:

$$\boldsymbol{G} = [\boldsymbol{G}^{(1)}, \boldsymbol{G}^{(2)}] \tag{3-1}$$

où $\boldsymbol{G}^{(1)}$ contient [(x+1) h] éléments et $\boldsymbol{G}^{(2)}$ contient [(h+1) o] éléments. Une direction de descente est évaluée indépendamment pour chaque niveau par:

$$\boldsymbol{S}^{(L)}(n) = -[\boldsymbol{H}^{(L)}(n)]^{-1} \quad \boldsymbol{G}^{(L)}(n) \tag{3-2}$$

de telle sorte que la direction globale est définie par:

$$\boldsymbol{S} = [\boldsymbol{S}^{(1)}, \boldsymbol{S}^{(2)}] \tag{3-3}$$

La variation des poids du réseau est toujours définie par:

$$\Delta \boldsymbol{W}(n) = \lambda(n)\, \boldsymbol{S}(n) \tag{3-4}$$

Les deux matrices hessiennes $\boldsymbol{H}^{(L)}$ sont symétriques et de dimension [(x+1) h] et [(h+1) o] pour le premier et le deuxième niveau respectivement.

Définissons $\boldsymbol{G}_{j'}^{(L)}(n)$ comme le vecteur gradient dont les éléments sont tous les $\boldsymbol{G}_{i'j'}^{(L)}(n)$ connectés au même neurone et où :

$$\boldsymbol{G}^{(1)} = [\boldsymbol{G}_1^{(1)}, \boldsymbol{G}_2^{(1)}, \ldots, \boldsymbol{G}_{j'}^{(1)}, \ldots, \boldsymbol{G}_h^{(1)}]$$

$$\boldsymbol{G}^{(2)} = [\boldsymbol{G}_1^{(2)}, \boldsymbol{G}_2^{(2)}, \ldots, \boldsymbol{G}_{j'}^{(2)}, \ldots, \boldsymbol{G}_o^{(2)}] \tag{3-5}$$

Le gradient global du réseau est une combinaison des deux gradients $\boldsymbol{G}^{(1)}$ et $\boldsymbol{G}^{(2)}$ comme à l'équation 3-1. Les gradients $\boldsymbol{G}_{j'}^{(1)}$ contiennent (x+1) éléments alors que ceux de $\boldsymbol{G}_{j'}^{(2)}$ en contiennent (h+1). Une direction de descente est évaluée indépendamment pour chaque neurone par :

$$\boldsymbol{S}_{j'}^{(L)}(n) = -[\boldsymbol{H}_{j'}^{(L)}(n)]^{-1} \quad \boldsymbol{G}_{j'}^{(L)}(n) \tag{3-6}$$

Pour la méthode BFGS-N, la direction globale de descente $\boldsymbol{S}$ est constituée de l'assemblage de directions de descente correspondantes aux neurones dans le réseau. L'ajustement des poids du réseau s'effectue par l'équation (3-4).

Pour la méthode BFGS-M, la direction globale de descente, $\boldsymbol{S}$, est construite par une combinaison des deux méthodes précédentes : un vecteur $\boldsymbol{G}^{(1)}$ qui a pour éléments tous les $\boldsymbol{G}_{i'j'}^{(1)}$ du premier niveau et un vecteur $\boldsymbol{G}^{(2)}$ tel que défini à l'équation (3-5).

Un des avantages de ces méthodes est que les approximations des matrices hessiennes peuvent être calculées indépendamment et donc en parallèle.

3.3 Comparaison de différentes méthodes d'optimisation

Pour évaluer la performance des modifications proposées, six algorithmes sont comparés dans quatre exemples. Les six algorithmes sont rétro-propagation (RP), le gradient conjugué (GC), la méthode quasi-Newton avec une mise à jour BFGS et les méthodes quasi-Newton modifiées avec respectivement les mises à jour BFGS-L (la mise à jour en fonction des niveaux), BFGS-M (la mise à jour en fonction des niveaux cachés et des neurones de sorties) et BFGS-N (la mise à jour en fonction des neurones). L'algorithme rétro-propagation utilise l'ajustement

périodique des poids, présenté à l'équation 2-42. L'algorithme du gradient conjugué est celui de Fletcher et coll. [1964] avec une réinitialisation de la direction de descente à tous les (t+1) itérations où t est le nombre de variables à minimiser. Le pas de descente $\lambda(n)$ est évalué avec une précision relative de 10^{-9} à l'aide de la méthode de recherche unidirectionnelle du nombre d'or (Edgar et coll. [1988]). Pour les algorithmes quasi-Newton et quasi-Newton modifiés, le pas de descente est évalué par une recherche unidirectionnelle qui respecte le critère d'Armijo (Armijo [1966]). De plus, il n'y a pas de correction appliquée lorsque les matrices hessiennes ne sont pas définies positives, une procédure qui consomme trop de temps de calcul. A la place, lorsque les matrices hessiennes approximées deviennent non-définies positives, elles sont remplacées par la matrice identité. Une fonction d'activation sigmoïde (équation 2-27) est utilisée pour tous les réseaux de neurones. Pour effectuer les comparaisons entre les différents algorithmes, dix ensembles de poids initiaux sont générés aléatoirement et les six algorithmes utilisent chacun des ensembles initiaux. Les calculs sont faits avec un ordinateur Unix IBM RS/6000 à l'exception du premier exemple, le test XOR, qui est fait avec un ordinateur PC-486 66 Mhz. Les algorithmes programmés pour le système IBM RS/6000 appellent les sous-programmes d'algèbre linéaire BLAS qui permettent d'augmenter considérablement les performances des opérations algébriques pour les grands systèmes linéaires. La codification interne de ces sous-programmes est spécifique à chaque plate-forme et est optimisée pour obtenir les meilleures performances possibles en fonction des caractéristiques de l'architecture de l'ordinateur.

3.3.1 Exemple du XOR

Le premier exemple est le test du « ou exclusif » connu sous le nom anglais de XOR. Cet exemple est devenu célèbre à la fin des années soixante lorsqu'un article de Minsky et coll [1969] a démontré que les réseaux de neurones de l'époque étaient incapables de classer des systèmes fort simples comme le « ou exclusif ». Cet article a eu pour effet de ralentir considérablement les recherches dans le domaine des réseaux de neurones, et il fut tout à fait normal lors de la résurgence des activités de recherche au début des années quatre-vingt de s'attaquer prioritairement à la résolution de ce problème.

C'est un exemple de classification qui consiste à apprendre la table de vérité du « ou exclusif ». Le tableau 3.1 présente les valeurs utilisées pour l'apprentissage. Afin de ne pas saturer la fonction sigmoïde, les valeurs de sorties Y dans la table de vérité sont remplacées par [0.1] et [0.9].

TABLEAU 3.1 Valeur de la table de vérité du XOR

X(1)	X(2)	Y
0	0	.9
0	1	.1
1	0	.1
1	1	.9

Le réseau de neurones utilisé possède deux neurones dans la couche cachée. Dans cet exemple, un ensemble de poids $\boldsymbol{W}$ est considéré une solution lorsque la valeur de la fonction objective E est inférieure à 10^{-5}. L'ensemble de poids à ajuster comprend 9 éléments. Pour l'algorithme de rétro-propagation, la vitesse d'apprentissage η et le momentum α sont fixés respectivement à 0.7 et 0.9. Comme cet exemple n'a qu'un neurone à la couche de sortie, les algorithmes BFGS-L et BFGS-M sont identiques.

Le tableau 3.2 rapporte le nombre de cas qui ont convergé à partir des dix ensembles initiaux de poids, le nombre d'itérations, le temps de calcul et le nombre d'éléments des matrices hessiennes. Les cas pour lesquels le critère d'arrêt n'est pas rencontré après 2000 itérations sont considérés comme ne convergeant pas et sont exclus des statistiques.

TABLEAU 3.2 Résultats du problème du XOR

Méthode	Convergé	Itérations	Temps		Matrices
$w = 9$	/10	moyenne	moyenne	relatif	nombre d'éléments
BP	10	305	6.34	3.48	
GC	5	71	1.82	1.0	
BFGS	5	124	11.39	6.26	81
BFGS-L	6	64	5.51	3.03	45
BFGS-M	6	64	5.51	3.03	45
BFGS-N	9	87	6.45	3.54	27

Lorsque l'on compare le nombre d'itérations les méthodes GC, BFGS-L (BFGS-M) et BFGS-N ont des résultats similaires. Par contre, le temps requis pour une itération est beaucoup plus faible pour la méthode du gradient conjugué et cette dernière obtient donc des temps de calcul de 3 à 3.5 fois inférieure. L'algorithme de rétro-propagation a des résultats comparables en terme de temps calcul à ceux obtenus par les méthodes BFGS-L et BFGS-N bien qu'il demande quatre fois plus d'itérations. L'algorithme BFGS n'est pas performant lorsque l'on compare le nombre d'itérations ou le temps de calcul. Les mauvaises performances de l'algorithme BFGS pour cet exemple proviennent du fait que l'approximation de la matrice hessienne est souvent non-définie positive. Dans ces situations, la matrice hessienne est remplacée par la matrice identité, ce qui réduit la méthode quasi-Newton à une méthode de gradient simple et augmente donc le nombre d'itérations nécessaire sans pour autant diminuer la charge de calcul pour chaque itération.

La comparaison du nombre de cas convergés est curieuse. Plusieurs méthodes ont obtenue un très faible pourcentage de cas convergés et seul la méthode de rétro-propagation et la méthode BFGS-N ont des résultats satisfaisants. Cela est en grande partie causé par la faible

valeur de la fonction E qui est retenue pour considérer qu'un ensemble de poids converge vers une solution et que cet exemple est truffé de minimums locaux qui sont situés à une valeur de la fonction objective supérieure au critère d'arrêt. Pour ce réseau de neurones de petite taille la réduction du nombre d'éléments des matrices hessiennes (de 81 à 27) n'est pas appréciable.

3.3.2 Exemple de diagnostics de pannes

Le deuxième exemple est tiré des travaux d'Hoskin et coll. [1988] qui ont utilisé un réseau de neurones pour diagnostiquer des pannes dans une série de trois de réacteurs CSTR dans lesquels a lieu une réaction du second ordre $A \Rightarrow B$. Les trois réacteurs opèrent de façon isotherme et à un débit volumique constant. Six variables d'état du système sont utilisées comme indicateurs de pannes à l'entrée du réseau : débit volumique, température et concentration des composés A et B à l'entrée et à la sortie des réacteurs. Six pannes possibles qui impliquent le débit volumique, la température et la concentration du composé A à l'entrée des réacteurs correspondent à la sortie du réseau de neurone. Le tableau 3.3 présente l'ensemble d'entraînement des douze modèles d'Hoskin et coll. [1988] .

TABLEAU 3.3 Ensemble d'entraînement pour le diagnostic de pannes

Entrée						Sortie					
F (ft^3 min^{-1})	T °F	C_A^{in} (lbmol ft^{-3})	C_B^{in} (lbmol ft^{-3})	C_A^{out} (lbmol ft^{-3})	C_B^{out} (lbmol ft^{-3})	C_A^{in}		F		T	
						bas	haut	bas	haut	bas	haut
18.0	190	0.3	3.18	0.2275	3.252	0.9	0.1	0.1	0.1	0.1	0.1
18.0	190	0.6	2.88	0.3755	3.104	0.9	0.1	0.1	0.1	0.1	0.1
18.0	190	1.3	2.18	0.5990	2.881	0.1	0.9	0.1	0.1	0.1	0.1
18.0	190	1.6	1.88	0.6681	2.812	0.1	0.9	0.1	0.1	0.1	0.1
13.0	190	1.0	2.48	0.4475	3.033	0.1	0.1	0.9	0.1	0.1	0.1
15.0	190	1.0	2.48	0.4777	3.002	0.1	0.1	0.9	0.1	0.1	0.1
22.0	190	1.0	2.48	0.5600	2.920	0.1	0.1	0.1	0.9	0.1	0.1
26.0	190	1.0	2.48	0.5958	2.884	0.1	0.1	0.1	0.9	0.1	0.1
18.0	150	1.0	2.48	0.8960	2.584	0.1	0.1	0.1	0.1	0.9	0.1
18.0	210	1.0	2.48	0.3170	3.170	0.1	0.1	0.1	0.1	0.1	0.9
18.0	230	1.0	2.48	0.1703	3.310	0.1	0.1	0.1	0.1	0.1	0.9
18.0	190	1.0	2.48	0.5190	2.961	0.1	0.1	0.1	0.1	0.1	0.1

Pour cet exemple, les valeurs de sortie sont fixées à [0.9] ou [0.1] selon qu'elles représentent une forte ou une faible probabilité. Leonard et coll. [1990] ont utilisé le même exemple pour comparer l'efficacité de l'algorithme de gradient conjugué par rapport à l'algorithme de rétro-propagation utilisé par Hoskin et coll. [1988]. Les résultats ont démontré que le gradient conjugué ne constitue pas une réelle amélioration lorsque le temps de calcul est comparé. Les résultats des deux groupes ont montré que, pour l'algorithme rétro-propagation, le nombre d'itérations minimal est obtenu avec une configuration de six neurones dans la couche

cachée et des valeurs de η et α fixées respectivement à 0.65 et 0.9 . Ces valeurs seront donc utilisées dans la présente étude. Le critère d'arrêt utilisé par Hoskin et coll. [1988] et par Leonard et coll. [1990] est le suivant : l'erreur absolue sur toutes les valeurs de l'ensemble d'entraînement est inférieure à 10^{-1} ou la fonction E change par un facteur de moins de 10^{-7} entre deux itérations successives. Dans ces conditions, les auteurs ont obtenu respectivement des moyennes sur dix essais de 34 et 31.5 itérations. En utilisant les mêmes critères d'arrêt nous avons obtenu une moyenne de 30.4 itérations. Leur critère d'arrêt est inapproprié pour comparer différents algorithmes sur une base de temps de calcul, et il est remplacé dans cette étude par un critère global qui consiste à minimiser la fonction objective E à une valeur inférieure à 10^{-1}. Le nombre de poids w que contient le réseau est de 84. Les résultats des différents algorithmes pour cet exemple sont présentés au tableau 3.4

TABLEAU 3.4 Résultats du problème de diagnostics de pannes

Méthode	Convergé	Itérations	Temps		Matrice
$w = 84$	/10	moyenne	moyenne	relatif	nombre d'éléments
BP	9	1748	8.35	1.63	
GC	7	228	7.61	1.49	
BFGS	9	172	7.13	1.39	7056
BFGS-L	9	443	11.67	2.30	3528
BFGS-M	7	264	5.12	1.0	2058
BFGS-N	9	696	8.12	1.59	588

Les résultats montrent qu'à l'exception de l'algorithme BFGS-L, toutes les méthodes sont supérieures à l'algorithme rétro-propagation si l'on compare le temps de calcul ou le nombre d'itérations. Sur une base de temps de calcul, l'algorithme le plus performant est BFGS-M alors que les méthodes BFGS, BFGS-N, GC et BP ont des temps relatifs qui varient de 1.39 à 1.63. De

plus, il demande d'emmagasiner seulement le tiers des éléments de la matrice hessienne complète. Lorsque l'ensemble des interactions du deuxième ordre est considéré pour les poids de la couche finale (méthodes BFGS-L vs. BFGS-M), le nombre d'itérations double (443 vs. 264). Ce résultat a priori surprenant n'a été observé dans aucun autre exemple. Il est associé à la procédure de mise à jour de la matrice hessienne approximée par la méthode BFGS. Dans un réseau de neurones, Bishop [1992] a montré que la matrice hessienne est creuse pour les interactions du deuxième ordre entre les poids de différents neurones de la couche finale. Les méthodes quasi-Newton, telle BFGS, ne respectent pas nécessairement cette propriété du réseau en approximant la matrice hessienne et peuvent introduire artificiellement des interactions entre ces neurones. Ce phénomène est sûrement amplifié lorsque l'on ne considère pas l'ensemble des interactions comme dans la méthode BFGS-L et cela peut, dans certains cas, modifier suffisamment la nature de la matrice hessienne approximée pour la rendre semi-définie positive et forcer le remplacement de l'approximation par la matrice identité. C'est ce qui s'est produit dans cet exemple de sorte que l'algorithme BFGS-L se comporte comme une méthode de descente directe qui demande beaucoup plus d'itérations.

3.3.3 Exemple du bioréacteur

Le troisième exemple consiste en la modélisation d'un système à la dynamique complexe. Il s'agit d'un réacteur bien mélangé à alimentation continue (CFSTR) dans lequel se produit une culture de cellules dont la croissance est inhibée par une trop grande concentration de substrat. Agrawal et coll. [1982] présentent le modèle de ce système ainsi que l'analyse de la dynamique. Ce même exemple est utilisé au chapitre 5 comme banc d'essai du contrôleur adaptatif par entraînement spécialisé.

Les équations du bilan de masse écrites sous une forme adimensionnelle sont données par :

$$\frac{dC_1}{d\tau} = -C_1 + Da \;\; C_1 \;\; (1 - C_2) \;\; e^{C_2/\gamma}$$

$$\frac{dC_2}{d\tau} = -C_2 + Da \;\; C_1 \;\; \frac{1+\beta}{1+\beta-C_2}(1-C_2) \;\; e^{C_2/\gamma}$$

et où C_1 et C_2 sont respectivement la masse adimensionnelle des cellules et la conversion de substrat, β est un paramètre relié au coefficient de rendement, γ est un paramètre relié au taux spécifique de croissance maximal, *Da,* la variable manipulée, est le nombre de Damkohler. Ce nombre est égal au rapport du taux spécifique de croissance aux concentrations d'entrée sur le taux de dilution dans le réacteur. Le temps de séjour adimensionnel τ est le produit du temps et du taux de dilution. Lorsque β est égal à 0.02 et γ est égal à 0.48, une bifurcation de Hopf se produit à un point *Da*=1.21 au-delà duquel le système se développe vers un cycle limite.

Le réseau de neurones est utilisé pour modéliser la dynamique du procédé dans le régime d'opération de *Da* appartenant à [1.2-2.0]. L'ensemble d'entraînement de quarante présentations a été développé en générant un signal aléatoire de la variable *Da* dans la région correspondante (figure 3.2) et en intégrant le modèle avec un pas d'intégration de $\tau = 0.1$, à partir d'un point initial *Da*=1.6 et des conditions d'équilibre correspondantes pour C_1 et C_2. La structure du réseau est faite : d'une couche de neuf entrées qui correspondent à *Da*, C_1 et C_2 aux temps (τ), (τ -*1*) et (τ -*2*) ; d'une couche cachée de trente neurones ; d'une couche de sortie de deux neurones qui prédisent C_1 et C_2 à (τ +*1*). Un ensemble de poids est considéré comme convergé lorsque l'erreur totale *E* est inférieure à 10^{-2}. Avec les 362 éléments qui le compose, ce réseau est de taille moyenne. Pour l'algorithme rétro-propagation, les meilleurs résultats ont été obtenus avec les paramètres suivant: η=0.7 et α=0.9 .

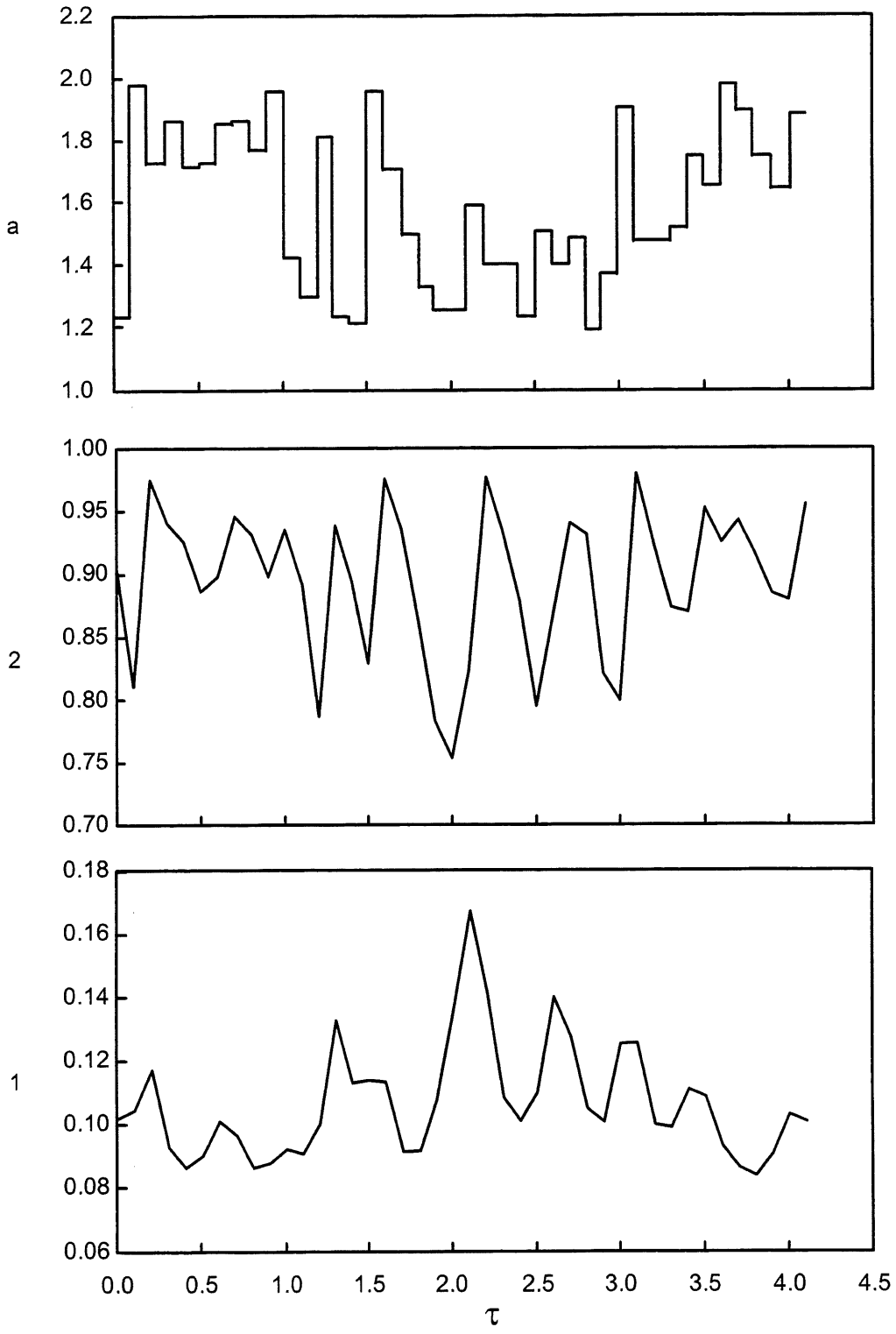


Figure 3.2 Graphiques des données d'entraînement pour le modèle du bioréacteur.

Le tableau 3.5 présente les résultats pour l'exemple du bioréacteur. Les essais pour lesquels les méthodes n'ont pas convergé en moins de 2000 itérations (20 000 pour BP) sont exclus des statistiques. L'analyse de la moyenne des itérations pour les méthodes quasi-Newton montre que plus il y a d'éléments inclus dans l'approximation de la matrice hessienne plus la méthode est performante. Toujours en fonction du nombre d'itérations, la méthode du gradient conjugué se situe entre les méthodes BFGS-M et BFGS-N. Comme le temps nécessaire pour effectuer une itération varie beaucoup d'une méthode à l'autre, les résultats les plus significatifs sont observés en comparant le temps moyen pour obtenir une solution. Le meilleur résultat provient de la méthode BFGS-N suivie de près de la méthode BFGS avec un temps relatif à la méthode précédente de 1.11 et ce bien que la méthode BFGS prenne en moyenne quatre fois moins d'itérations que la méthode BFGS-N. Toutes les méthodes quasi-Newton se sont avérées plus performantes que la méthode du gradient conjugué ou l'algorithme rétro-propagation qui ont des temps relatifs moyens de 3.6 et 6.4 respectivement. Dans la configuration du réseau de neurones utilisée, la réduction du nombre de termes inclus dans l'approximation de la matrice hessienne entre les méthodes BFGS et BFGS-N est très importante avec un rapport de 26.6. Les gains en espace pour les méthodes BFGS-L et BFGS-M sont plus marginaux.

TABLEAU 3.5 Résultats de l'exemple du bioréacteur

Méthode	Convergé	Itérations	Temps		Matrice
$w = 362$	/10	moyenne	moyenne	relatif	nombre d'éléments
BP	9	10 174	189.3	6.44	
GC	10	731	105.4	3.59	
BFGS	10	232	32.5	1.11	131 044
BFGS-L	10	425	45.7	1.55	93 844
BFGS-M	10	553	59.4	2.02	91 922
BFGS-N	9	863	29.4	1	4 922

3.3.4 Exemple de l'analyse de spectres d'absorptiométrie ultraviolette multi-longueur d'onde

L'analyse spectrale est un domaine d'étude qui pose des défis intéressants pour les réseaux de neurones. Leurs grands nombres d'éléments à analyser et leurs caractéristiques hautement non linéaires en font des exemples tout à fait appropriés pour la comparaison des méthodes d'entraînement des réseaux de neurones. Dans cet exemple, un réseau de neurones est utilisé pour corréler des spectres d'absorptiométrie ultraviolette multi-longueur d'onde d'eaux usées à des paramètres de qualité de l'eau comme les solides en suspension (SS) et la demande chimique en oxygène (DCO). Thomas et coll. [1990] ont démontré que plusieurs paramètres de qualité de l'eau sont reliés au spectre l'absorptiométrie ultraviolette dans les longueurs d'onde de 200 à 350 nm. Ces spectres sont hautement influencés par la présence de composés chimiques dans les eaux tels les nitrates, le chrome et les phénols. Par une analyse mathématique des spectres, Thomas et coll. [1990] sont arrivés à prédire, avec une bonne précision, la concentration de ces produits. Plus récemment Renard [1995] a étudié l'utilisation de réseaux de neurones pour produire un modèle qui estime, à partir d'un spectre d'absorptiométrie ultraviolette, des paramètres de qualité des eaux comme la demande biologique en oxygène (DBO), la DCO, le carbone organique total (COT) et les solides en suspension. Ces réseaux de neurones sont de très grande dimension puisque le spectre utilisé à l'entrée est parfois discrétisé en 150 longueurs d'onde.

La figure 3.3 présente deux courbes types utilisées dans cette étude correspondant aux valeurs de SS et de DCO associées. Le réseau de neurones contient cinquante et une entrées, chacune représentant l'absorptiométrie à une longueur d'onde du spectre 200-350 nm, vingt neurones cachés et deux neurones de sortie indiquant la mesure de solides en suspension et la mesure de DCO des échantillons d'eau. Ce réseau de grande taille contient 1082 éléments à ajuster. L'ensemble d'entraînement est fait de quinze spectres qui couvrent des plages de SS de 8 à 436 mg/l et de DCO de 25 à 888 mg/l O_2. Les calculs sont arrêtés lorsque l'erreur totale E est inférieure à 10^{-3}. La vitesse d'apprentissage η et le momentum α pour l'algorithme de rétro-propagation sont fixés à 0.3 et 0.7 respectivement.

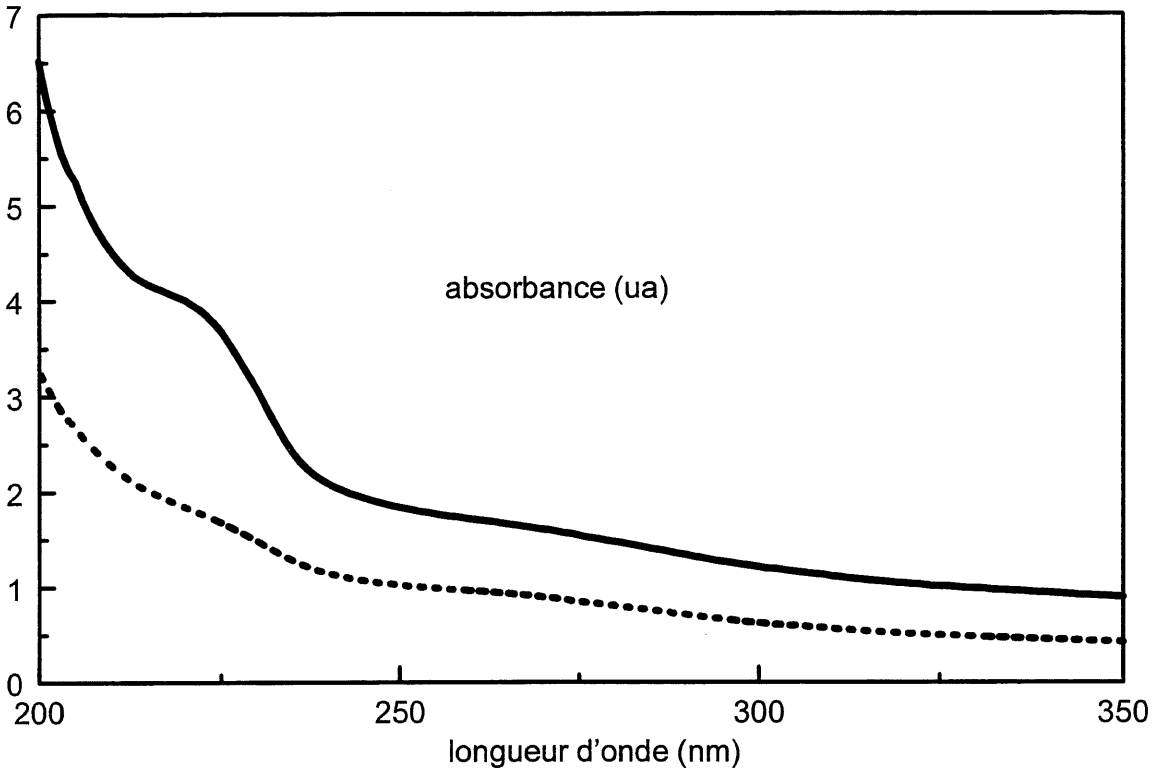


Figure 3.3 Exemple de deux spectres d'absorption utilisés pour l'identification des solides en suspension (SS) et de la demande chimique en oxygène (DCO) d'eaux usées.
——— : SS = 182 mg/l ; DCO = 633 mg/l O_2
- - - - - : SS = 68 mg/l ; DCO = 199 mg/l O_2

Les résultats de ces essais sont présentés au tableau 3.6. Les cas qui n'ont pas convergé en moins de 3 000 itérations (50 000 pour BP) sont exclus des statistiques. Le nombre d'itérations moyen nécessaire pour les trois variantes de la méthode quasi-Newton proposées sont sensiblement les mêmes et sont près du double du nombre d'itérations que demande la méthode BFGS. Les méthodes du gradient conjugué et de rétro-propagation nécessitent passablement plus d'itérations. Lorsque les temps de calculs sont comparés, c'est la méthode BFGS-N qui est de loin la plus rapide avec un temps relatif 2.5 fois inférieur à la méthode du gradient conjugué. Pour les trois méthodes qui emmagasinent des approximations de la matrice hessienne considérable, BFGS, BFGS-L et BFGS-M, le temps de calcul par itération est important puisque la gestion de la mémoire pour les grandes matrices diminue de façon importante l'efficacité des calculs même lorsque les programmes emploient des sous-programmes performants de calcul

algébrique. Il est intéressant de constater que, dans cet exemple, l'algorithme rétro-propagation n'est pas beaucoup plus lent que les méthodes BFGS-L et BFGS-M. En ce qui a trait à l'espace mémoire utilisé avec la méthode BFGS-N, il est d'environ un vingtième de celui que nécessitent les autres méthodes quasi-Newton essayées.

TABLEAU 3.6 Résultats de l'exemple de l'analyse de spectres d'absorptiométrie ultraviolette multi-longueur d'onde

Méthode	Convergé	Itérations	Temps		Matrice
$w = 1082$	/10	moyenne	moyenne	relatif	nombre d'éléments
BP	9	28 671	525.2	13.5	
GC	10	1769	96.8	2.49	
BFGS	9	249	237.2	6.10	1 170 724
BFGS-L	9	406	416.0	10.69	1 083 364
BFGS-M	8	450	450.1	11.78	1 082 482
BFGS-N	10	479	38.9	1	54 962

3.3.5 Analyse des performances des nouvelles méthodes

Trois modifications à l'approche quasi-Newton classique, les méthodes BFGS-L, BFGS-M et BFGS-N, ont été présentées aux sections précédentes. Ces différents exemples ont montré que les hypothèses supportant ces méthodes sont justes et que les résultats obtenus en terme de temps de calcul rendent ces approches attrayantes. Lorsque les réseaux de neurones à entraîner sont de taille moyenne ou grande ($w > 300$), la méthode BFGS-N, qui considère uniquement les interactions du deuxième ordre pour les poids arrivant à un même neurone, est plus performante que la méthode BFGS, la méthode du gradient conjugué et l'algorithme rétro-propagation. Elle apporte une amélioration majeure en terme de temps de calcul. Ce gain est obtenu sans une

augmentation majeure de l'espace mémoire requis par rapport aux méthodes de gradient conjugué et rétro-propagation. De plus, il n'y a pas de paramètres de convergence à ajuster comme dans l'algorithme de rétro-propagation. A l'exception de l'exemple de diagnostic de pannes dans trois réacteurs en série où la méthode BFGS-M s'est avéré le plus rapide, les méthodes BFGS-L et BFGS-M, dans les configurations de réseaux de neurones essayées, ne présentent pas un compromis intéressant en terme d'espace mémoire et de temps de calcul. De fait, la méthode BFGS-L semble de peu d'intérêt puisque l'algorithme BFGS-M représente toujours un meilleur compromis. Cette dernière contient une structure pour la couche finale ayant moins d'éléments.

4. CONTRÔLE DE PROCÉDÉ PAR RÉSEAUX DE NEURONES

4.1 Introduction

Ce chapitre présente une revue des différentes approches utilisées pour trouver un modèle dynamique d'un procédé par des réseaux de neurones. Vient ensuite une revue des techniques de contrôle de procédés de génie chimique qui utilisent des réseaux de neurones. Cette revue se limite aux procédés qui sont continus. La dernière partie présente l'algorithme de contrôle adaptatif par entraînement spécialisé de réseaux de neurones, aussi appelé contrôleur neuronal adaptatif.

4.2 Revue de l'utilisation de réseaux de neurones pour l'identification d'un procédé

L'identification d'un procédé est la procédure par laquelle la dynamique d'un système est modélisée à partir d'un ensemble de données composé de valeurs d'entrée-sortie du système. L'étape d'identification est une composante importante du design des politiques de contrôle de procédé. De plus, plusieurs algorithmes de contrôle utilisent directement un modèle de la dynamique du procédé. Plusieurs travaux ont montré que les réseaux de neurones à propagation avant sont un outil de modélisation non linéaire tout à fait indiqué pour procéder à l'identification d'un système (Narendra et coll. [1990], Bhat et coll. [1990], Ydstie [1990], Thibault [1991], Nahas et coll. [1992], Latrille et coll. [1994]).

Les réseaux de neurones peuvent être vus comme des approximateurs non linéaires universels (Hecht-Nielsen [1989]). Pour l'identification d'un procédé, ils sont utilisés pour établir la relation entre les sorties d'un système à un temps présent et les entrées et sorties aux temps précédents. De par leur structure, les réseaux de neurones établissent le « modèle » et ses paramètres lors de la phase d'apprentissage. Pour les systèmes dont la dynamique est complexe, il est souvent nécessaire en travaillant avec des techniques comme NARMAX d'utiliser plusieurs modèles (Chen et coll. [1989]) pour couvrir l'ensemble du domaine dynamique alors qu'un seul réseau avec suffisamment de neurones dans la couche cachée est toujours adéquat. Comme les réseaux de neurones sont hautement non linéaires, ils sont par ailleurs en mesure de capter

beaucoup plus la dynamique d'un procédé que les méthodes basées sur des techniques de convolution linéaire ou les techniques ARMA « autoregressive moving average ».

Plusieurs travaux ont comparé la qualité de l'identification d'un système obtenue par un réseau de neurones à celle d'autres approches. Des comparaisons avec la méthode ARMA ont été faites par Bhat et coll. [1990] et Psichogios et coll. [1991] et avec la méthode NARMAX par Chen et coll. [1990], Thibault [1991] et Seborg [1994]. Dans tous ces travaux, les réseaux de neurones se sont avérés plus performants en terme de la qualité du modèle dynamique qu'ils fournissent.

Un modèle dynamique qui prédit l'état du système à la prochaine période d'échantillonnage est appelé « modèle prédictif à un pas en avant ». Les réseaux de neurones à propagation avant entièrement connectés, tels que définis à la Figure 2.1, sont tout à fait adaptés pour établir des modèles prédictifs à un pas en avant. Sur la Figure 4.1 on peut voir la configuration générale de la modélisation directe de la dynamique d'un procédé qui a une variable mesurée y et une variable manipulée u (système S.I.S.O. « single input and single output ») . Le vecteur d'entrée du réseau est composé de valeurs passées d'entrées et de sorties du système alors que le vecteur de sortie prédit l'état du système au prochain pas en temps. La variable d est le nombre de pas en temps qui couvre le retard du procédé, $(n+1)$ représente le nombre d'échantillons de la variable de sortie du procédé, et $(m+1)$ est le nombre d'échantillons de la variable manipulée. La fonction F établit la correspondance entre l'entrée et la sortie du réseau et est définie par:

$$\hat{y}(t+1) = F\left[y(t),\ldots,y(t-n),u(t-d),\ldots,u(t-d-m)\right] \tag{4-1}$$

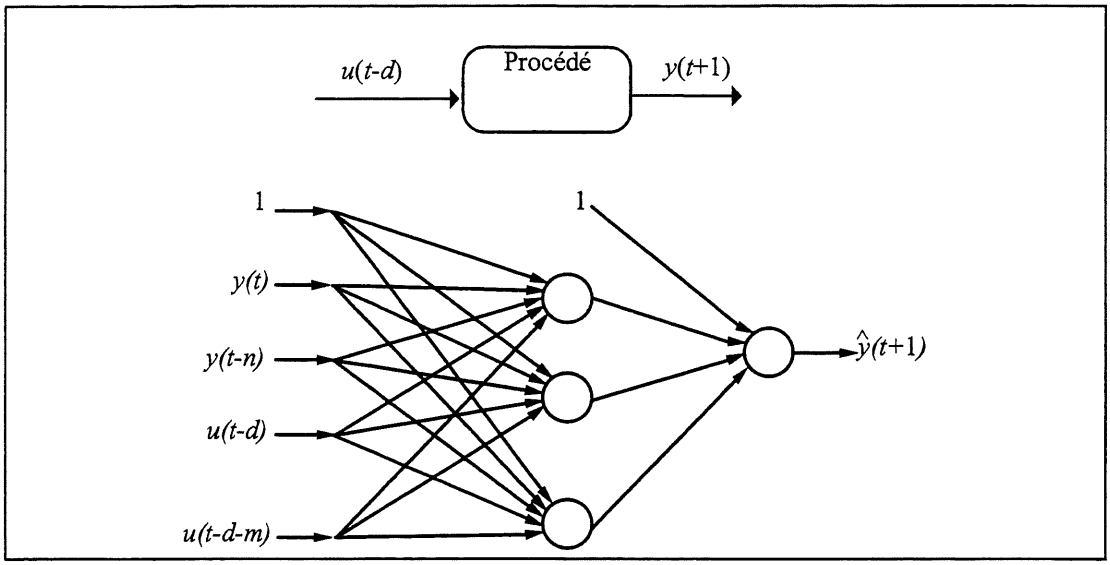


Figure 4.1 Schéma général de la modélisation directe de la dynamique d'un procédé. d est le nombre de pas en temps qui couvre le délai du procédé, $(n+1)$ est le nombre d'échantillons de la variable de sortie du procédé ; et $(m+1)$ est le nombre d'échantillons de la variable manipulée.

L'ensemble d'entraînement est construit à partir d'une fenêtre d'observation (Figure 4.2) glissant sur les évolutions dynamiques enregistrées. Cette fenêtre est appelée l'horizon d'observation. Le choix des valeurs d,m, et n est dicté par la nature du système à modéliser. Le choix de d est souvent le plus simple à évaluer puisque l'on a en général une bonne idée du délai inhérent à un procédé; par contre il n'existe pas de méthode sûre pour trouver la meilleure combinaison de m et de n. On peut toutefois utiliser les résultats qui proviennent des approches linéaires ARMA qui sont basées sur l'hypothèse qu'un système S.I.S.O. non linéaire de N variables d'états peut être reconstruit à partir de N valeurs d'entrées et de sorties, soit $m+1 = n+1 = N$ (Narendra et coll. [1990]).

L'exemple qui est illustré se rapporte aux situations qui sont fréquentes en contrôle de procédés où seules les variables contrôlées et les variables manipulées sont mesurées. Dans les cas où d'autres variables mesurées ayant une relation avec la dynamique du système sont disponibles, il peut être très avantageux de les inclure dans le modèle.

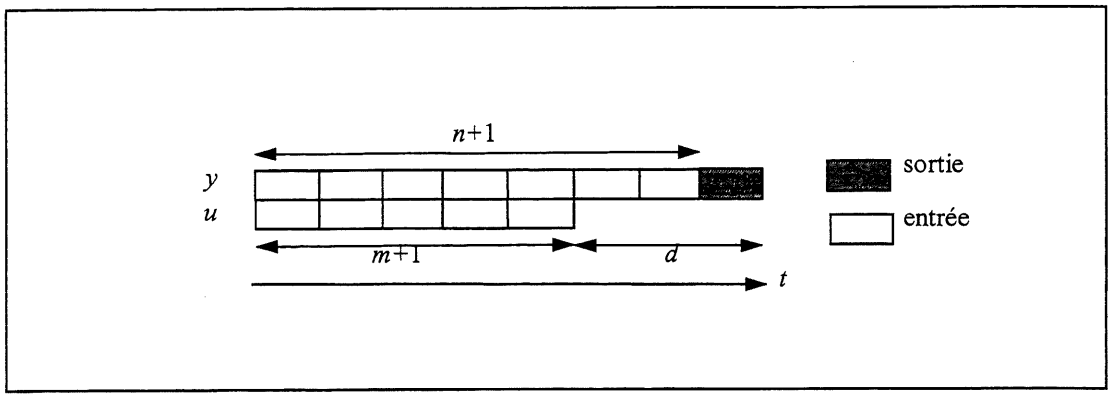


Figure 4.2 Fenêtre d'observation pour constituer les valeurs d'entraînement

Pour constituer le fichier d'entraînement, il est très important d'avoir des valeurs qui reflètent le plus possible le domaine et la dynamique dans lesquels le système va évoluer; par exemple s'assurer de bien couvrir toute la plage d'opération et d'avoir des variations entre deux itérations qui reflètent le comportement dynamique du procédé ; si la variable manipulée du contrôleur n'est pas autorisée à varier de plus de 20 %, cela devrait se refléter dans les valeurs d'entraînement. Les valeurs d'entraînement peuvent provenir de plusieurs sources comme d'une succession d'échelons en boucle ouverte, une variation aléatoire des variables manipulées ou la réponse d'un système à un signal P.R.B.S. (pseudo-random binary sequence).

Les réseaux de neurones à propagation avant peuvent aussi être entraînés pour établir le modèle inverse de la dynamique d'un système, c'est-à-dire déterminer à partir de l'état présent et passé du système quelle est la commande $u(t+1)$ qui amènera le système vers une consigne $y^*(t+d+1)$ (Figure 4.3). La variable d est le nombre d'itérations qui couvre le délai du procédé, $(n+1)$ représente le nombre d'échantillons de la variable de sortie du procédé, et (m) est le nombre d'échantillons de la variable manipulée. Une fonction G qui établit la correspondance entre l'entrée et la sortie du réseau peut donc être définie comme:

$$u(t) = G\left[y^*(t+d+1), y(t), \ldots, y(t-n), u(t-1), \ldots, u(t-m)\right] \qquad (4\text{-}2)$$

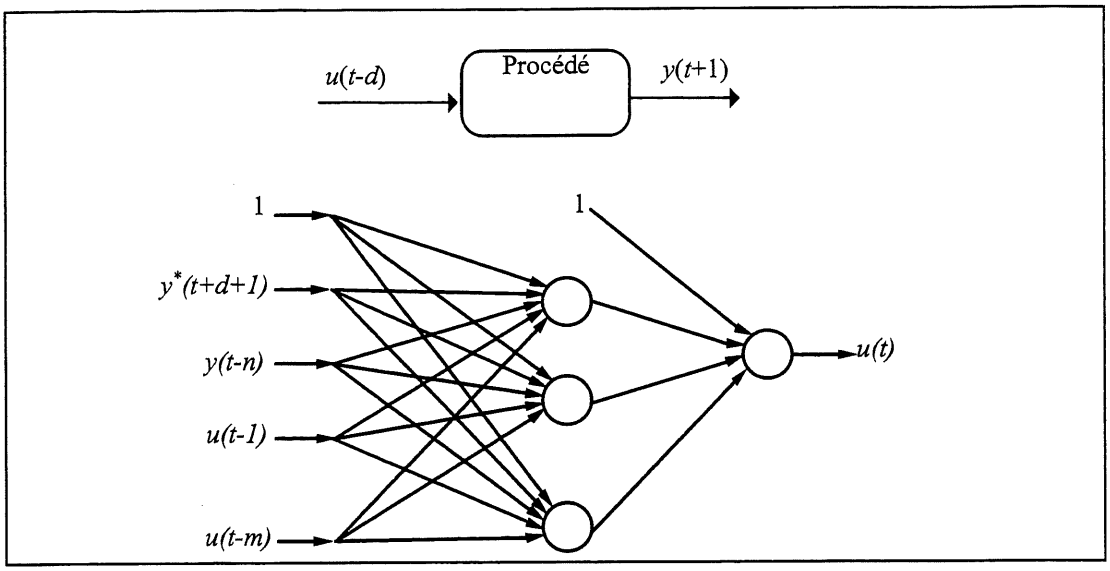


Figure 4.3 Schéma général de la modélisation de la dynamique inverse d'un procédé. $y^*(t+d+1)$ est la valeur désirée du procédé après un délai de d périodes d'échantillons ; $(n+1)$ est le nombre d'échantillons de la variable de sortie du procédé ; et (m) est le nombre d'échantillons de la variable manipulée.

L'ensemble d'entraînement pour les modèles inverses est aussi construit à partir d'une fenêtre d'observation et les mêmes soins doivent être apportés dans les méthodes utilisées pour obtenir ces valeurs. Psichogios et coll. [1991] notent que, de par la nature même des procédés, il est en général plus difficile de bien choisir l'approche pour constituer l'ensemble d'entraînement des modèles inverses que pour constituer l'ensemble d'entraînement des modèles directs.

Certaines situations demandent un modèle dynamique qui prédit le comportement du système pour plusieurs périodes d'échantillonnage à venir. C'est le cas lorsque l'on utilise un modèle dans une structure de contrôle prédictif à plusieurs pas en avant ou lorsque l'on veut contrôler l'état final d'un procédé batch ou semi-batch. Les réseaux de neurones à propagation avant peuvent être structurés de plusieurs façons pour établir ces modèles. L'approche la plus commune consiste à introduire dans le vecteur d'entrée du réseau de neurones les prédictions des itérations précédentes et à itérer le nombre de fois nécessaire pour couvrir la période désirée. Cette approche est souvent appelée « récurrence externe » (Su et coll. [1992]) puisqu'il n'y a aucun lien récurrent à l'intérieur du réseau de neurones. L'entraînement de ces réseaux est un peu délicat. L'approche de Narendra et coll. [1990] appelée « approche parallèle » est une technique

d'identification qui consiste à entraîner le réseau de la même façon dont il sera utilisé par la suite, c'est-à-dire par un ajustement continu des poids à chaque nouvelle présentation. L'entraînement des réseaux suivant l'approche parallèle pose souvent des difficultés. Selon Thibault et coll. [1992], l'expérimentation a montré qu'il est très difficile d'obtenir un modèle par l'approche parallèle à partir de poids initiaux déterminés aléatoirement. On peut facilement le concevoir car lors des itérations initiales d'optimisation des poids du réseau, les prédictions du modèle peuvent être très éloignées des valeurs souhaitées et ces mêmes prédictions sont réinsérées à l'entrée du réseau. Thibault et coll. [1992] proposent plutôt de faire une première optimisation en remplaçant dans le vecteur d'entrée du réseau les valeurs prédites par le modèle par des valeurs du système et d'utiliser les poids ainsi obtenus comme estimation de départ pour l'approche parallèle ; c'est l'approche que Narendra et coll. [1990] ont nommée « série-parallèle ». Latrille et coll. [1994] ont utilisé cette technique d'apprentissage avec succès pour la prédiction du temps final d'une fermentation de produit laitier.

Les réseaux de neurones récurrents sont une autre approche prometteuse pour établir des modèles prédictifs à plusieurs pas en avant. Ces réseaux diffèrent des réseaux de neurones à propagation avant entièrement connectés : ils incluent des liens pour la rétropropagation en plus des liens entre les neurones pour la propagation avant ; et certains neurones sont auto-excités, c'est-à-dire que la sortie est redirigée vers l'entrée. La description plus approfondie de ces modèles sort du cadre de notre étude, mais on peut trouver chez Su et coll. [1992] et chez You et coll. [1993] d'excellentes analyses.

4.3 Revue de littérature sur le contrôle de procédés à l'aide de réseaux de neurones

Une grande quantité d'articles publiés portent sur l'utilisation de réseaux de neurones dans des algorithmes de contrôle de procédés. Une revue exhaustive a été présentée par Thibault et coll. [1992]. Ceux qui sont présentés dans cette revue sont choisis de façon à illustrer les différentes tendances en mettant l'accent sur les applications reliées au génie chimique. Cette section est divisée en trois parties: la première présente les définitions des termes et des symboles utilisés ; la seconde étudie les algorithmes de contrôle utilisant les réseaux de neurones comme

modèle direct de la dynamique du procédé ; et la dernière présente des algorithmes de contrôle employant les réseaux de neurones comme modèle inverse c'est-à-dire produisant une commande directe au procédé.

4.3.1 Définitions des termes et des symboles

Sur la Figure 4.4, on peut voir le schéma d'un procédé et de son contrôleur. Le procédé a une variable mesurée y et une variable manipulée u (système S.I.S.O.) dont la valeur est déterminée par le contrôleur. La consigne du système y_{sp} est envoyée au contrôleur alors que les perturbations externes s'additionnent à la sortie du procédé pour produire l'état y. Dans une structure de coordonnées de temps discrètes, la consigne $y_{sp}(t)$ est envoyée au contrôleur et la commande $u(t)$ au procédé du temps (t) au temps $(t+1)$.

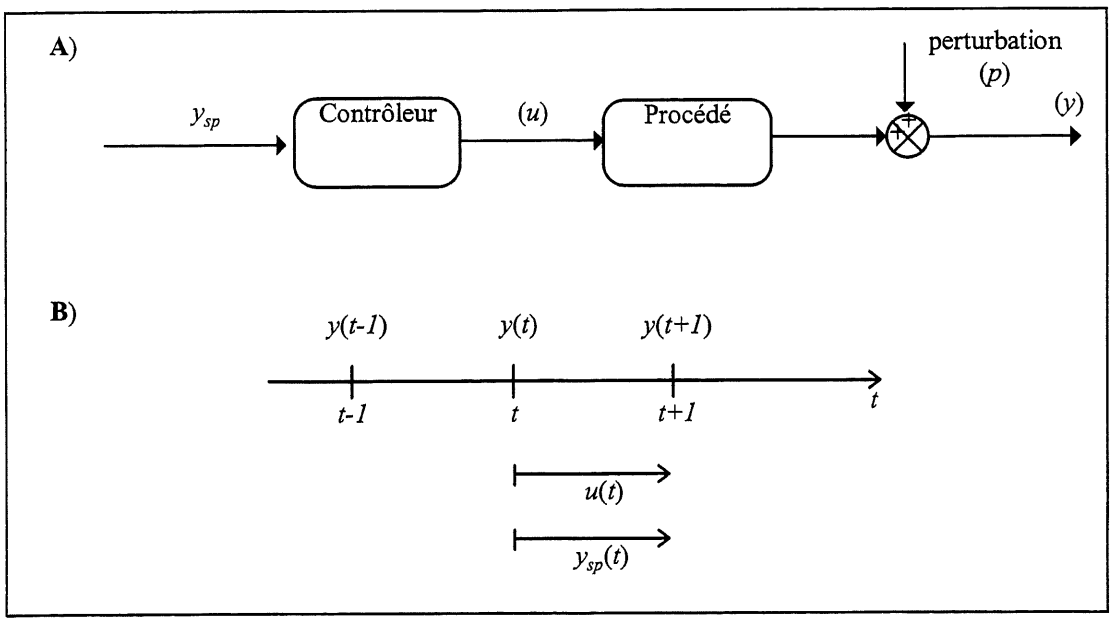


Figure 4.4 **A)** Définition des symboles d'une structure de contrôle S.I.S.O. ; y_{sp} est la consigne, u la commande du contrôleur, p une perturbation externe et y l'état du système.

B) Définition des coordonnées de temps pour les algorithmes dans le domaine discret.

Plusieurs approches modernes de contrôle de procédé, appartenant à la famille du contrôle par modèle prédictif, sont souvent utilisées en conjonction avec des réseaux de neurones. Les grandes lignes de cette famille de contrôleur ainsi qu'un des membres les plus populaires, le contrôle par modèle interne, seront donc brièvement présentés.

La stratégie de contrôle par modèle interne (IMC) a été formulée de façon rigoureuse par Garcia et coll. [1982] (Figure 4.5). Ils ont montré que, pour un système linéaire stable en boucle ouverte, le contrôleur idéal est tout simplement l'inverse du procédé. Le contrôleur ainsi calculé n'est pas toujours réalisable. Ils proposent donc de factoriser le modèle linéaire et de ne garder pour l'inversion que les termes qui ne contiennent pas de retards ou de zéro hors du cercle unitaire, donc les composantes qui sont en minimum de phase. Le contrôleur résultant est tout de même le meilleur contrôleur possible en fonction de la somme du carré des erreurs puisque les termes exclus du contrôleur sont inhérents à la physique du procédé et ne peuvent être éliminés par aucune procédure de contrôle. Pour ajouter de la robustesse au contrôleur, un filtre passe bas

est presque toujours ajouté à l'entrée. Economou et coll. [1986] ont étendu les résultats de Garcia et coll. [1982] pour les systèmes non linéaires. Ils ont démontré que les mêmes propriétés rendant le contrôle par modèle interne intéressant pour les systèmes linéaires se transposent pour les systèmes non linéaires. Principalement, un système stable en boucle ouverte peut être stabilisé en boucle fermée en construisant un contrôleur basé sur la pseudo-inverse du modèle non linéaire.

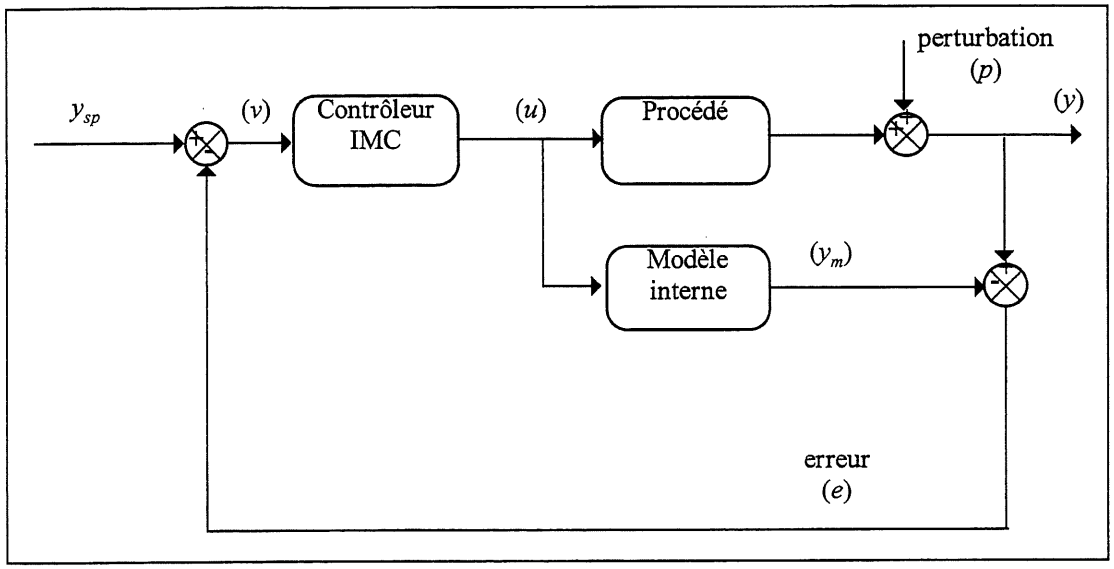


Figure 4.5 Schéma de la structure de contrôle par modèle interne (IMC)

Les méthodes regroupées dans la famille des contrôleurs par modèle prédictif (MPC) partagent toutes l'utilisation directe d'un modèle explicite identifié de façon indépendante à l'opération du contrôleur (Figure 4.6). Ce modèle est utilisé pour prédire l'impact des mouvements des variables manipulées sur l'état futur du système. Garcia et coll. [1989] font une excellente revue des principales méthodes qui peuvent être regroupées sous la dénomination de contrôle par modèle prédictif. Ces méthodes se distinguent selon qu'elles utilisent un modèle linéaire ou non linéaire, qu'elles contrôlent une ou plusieurs variables, qu'elles font des prédictions d'un ou de plusieurs pas de temps en avant, qu'elles incluent de façon implicite des contraintes sur les variables mesurées ou manipulées ou qu'elles minimisent une fonction objective provenant de la norme-1, de la norme-2 ou de la norme-∞. Les plus connues sont le contrôle par matrice dynamique (DMC) de Cutler et coll. [1979], le contrôleur prédictif non

linéaire multipas (MNPC) de Brengel et coll. [1989] et le contrôle par modèle interne (IMC) de Garcia et coll. [1982]. Garcia et coll. [1989] ont démontré que le contrôle par modèle interne est un cas particulier du contrôle par modèle prédictif et pouvait être reformulé sous la forme de contrôle prédictif à un pas sans contrainte.

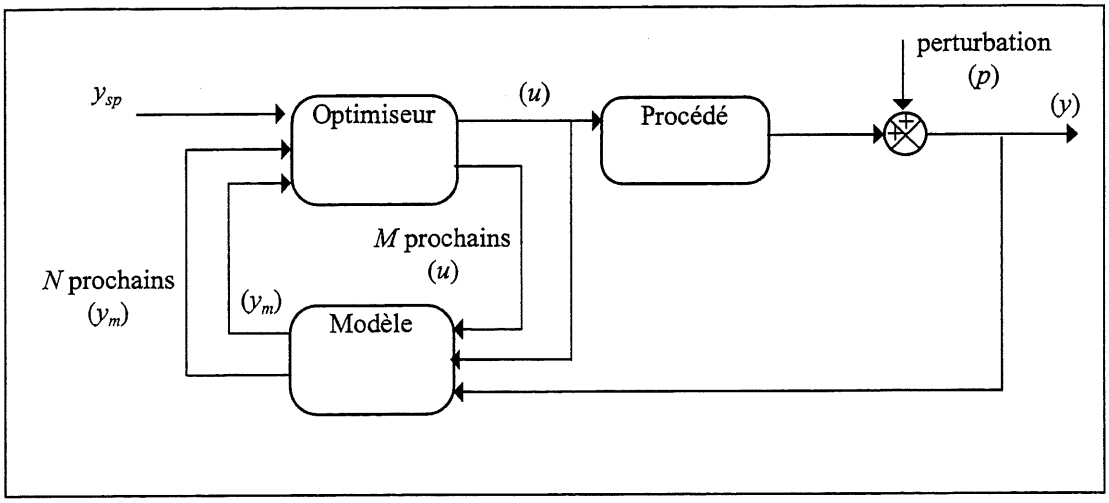


Figure 4.6: Schéma général de la structure des contrôleurs par modèle prédictif (MPC) ; M est l'horizon des variables manipulées et N l'horizon des variables contrôlées.

La plupart de ces techniques utilisent un horizon mobile de plusieurs pas d'échantillonnage en avant qui permet d'estimer la meilleure séquence de variables manipulées à implanter. A chaque période d'échantillonnage, tout le processus d'optimisation est effectué et la première commande évaluée est implantée. Les approches multipas ont l'avantage de pouvoir contrôler un système instable en boucle ouverte ou un système en non-minimum de phase, deux cas que les techniques prédictives à un pas ne peuvent pas traiter implicitement. Garcia et coll. [1982] ont montré qu'avec le contrôle par modèle interne, on doit d'abord stabiliser le système avec une boucle traditionnelle en rétroaction avant d'implanter le contrôleur IMC. De plus, on ne peut contrôler un système en non-minimum de phase avec un prédicteur à un pas que si le temps d'inversion du système est inférieur à la période d'échantillonnage.

Ces algorithmes utilisés en temps réel demandent une grande puissance de calcul et sont donc généralement limités aux procédés ayant de fortes valeurs ajoutées.

4.3.2 Contrôleur avec modèle direct du système

Ydstie [1990] utilise un réseau de neurones comme modèle dynamique du système dans une structure de contrôle prédictif à un pas en avant pour contrôler une simulation d'une réaction du deuxième ordre entre du thiosulfate de sodium et du peroxyde d'hydrogène dans un réacteur CSTR. Cette approche compare la consigne avec la prédiction du modèle et trouve par itération avec le réseau de neurones la commande qui amène le système au point de consigne. Parallèlement à la phase prédiction-contrôle, le réseau est entraîné en continu pour capter la dynamique du système, ce qui en fait une approche de contrôle adaptatif.

Dans les simulations présentées, la variable manipulée prend instantanément la valeur voulue. L'apprentissage est toujours fait à partir de la même séquence de variations de consigne. Comme il est aussi d'usage en contrôle adaptatif linaire, l'estimation du modèle est interrompue lorsque la variable manipulée atteint une contrainte (borne supérieure ou inférieure). Initialement, lors du démarrage de l'apprentissage du modèle, les performances sont similaires à celle d'un contrôleur adaptatif linéaire. Ydstie [1990] conclut que les performances ne sont bonnes qu'après une longue période d'apprentissage, c'est-à-dire lorsque le modèle est excellent dans la région d'opération. Le risque avec cette approche est qu'on doit recommencer l'apprentissage si la dynamique du système évolue dans le temps et que, pendant cette période, les performances du contrôleur ne seront pas satisfaisantes. De plus, tous les problèmes bien connus des réseaux de neurones comme le surentraînement et les minimums locaux sont toujours présents. Donc, si un point de consigne n'a jamais été demandé ou s'il ne l'a pas été depuis longtemps, il peut se passer un temps relativement long avant que tout écart au point de consigne soit éliminé. De plus, rien ne garantit que la consigne sera toujours atteinte exactement.

Psichogios et coll. [1991] ont étudié l'utilisation d'un réseau de neurones comme modèle non linéaire dans une structure de contrôle par modèle interne. Le contrôleur est calculé en inversant directement le réseau de neurones à chaque itération. Cette inversion est facilitée en utilisant les travaux de Jordan [1988] montrant que les réseaux de neurones à propagation avant

contiennent implicitement le jacobien de la sortie du réseau par rapport à l'entrée. Psichogios et coll. [1991] utilisent la méthode de Newton pour résoudre l'équation non linéaire suivante:

$$F(u,\boldsymbol{x}) - v = 0 \tag{4-3}$$

où $F(u,\boldsymbol{x})$ est la fonction de correspondance, établie par le réseau de neurones, qui génère la prédiction $\hat{y}$: l'entrée du contrôleur v est donnée par $v = y_{SP} - (y - \hat{y})$; et $\boldsymbol{x}$ est le vecteur d'entrée du réseau sans la commande u .

La solution de ce système donne la commande u qui amène la sortie du procédé y à la consigne y_{SP} (les termes $\hat{y}$ contenus dans l'équation (4-3) s'annulent). Les itérations de la méthode de Newton se font par l'équation:

$$u^{n+1} = u^n - \frac{F(u^n,\boldsymbol{x}) - v}{\partial(F(u^n,\boldsymbol{x}) - v)/\partial u^n} \tag{4-4}$$

qui est réécrite lorsque $\hat{y} = F(u^n,\boldsymbol{x})$ comme:

$$u^{n+1} = u^n - \frac{\hat{y} - v}{\partial \hat{y}/\partial u^n} \tag{4-5}$$

La méthode est utilisée pour le contrôle d'un réacteur CSTR non-isotherme avec une réaction réversible du premier ordre. Economou et coll. [1986] ont étudié ce même système pour lequel ils ont présenté la stratégie de contrôle par modèle interne non linéaire (NIMC). Les résultats obtenus par Psichogios et coll. [1991], lorsqu'ils considèrent que toutes les variables du système sont mesurées et intégrées au modèle par réseau de neurones, sont comparables à ceux présentés par Economou et coll. [1986] utilisant les équations exactes du système. Les auteurs concluent en exprimant certaines réserves quant à la difficulté associée à la solution de l'équation non linéaire par la méthode de Newton. En effet, la méthode peut ne pas converger en cas de mauvais estimés de départ, ou de systèmes où pour certains points de l'espace, les dérivées sont pratiquement nulles.

Nahas et coll. [1992] ont aussi utilisé un réseau de neurones comme modèle non linéaire dans une structure de contrôle par modèle interne très similaire à celle présentée parPsichogios et coll. [1991]. Toutefois, ils ont apporté deux modifications importantes à l'approche. La première concerne les conditions nécessaires pour garantir qu'il n'y aura pas d'écart à la consigne pendant l'opération du système en boucle fermée. Comme l'ont démontré Economou et coll. [1986], le contrôle par modèle interne non linéaire ne demande pas que le contrôleur soit l'inverse exact du modèle, mais il doit contenir l'inverse exacte du gain du système. Cette condition est essentielle pour garantir qu'il n'y aura pas d'écart à la consigne. Pour s'assurer que le gain du modèle par réseau de neurones est bien inversé, Nahas et coll. [1992] proposent d'utiliser les valeurs présentes et passées prédites par le modèle plutôt que celles du procédé, dans l'algorithme de Newton.

La deuxième modification apportée est l'ajout de façon explicite d'un filtre à l'entrée du contrôleur, une approche qui est tout à fait habituelle dans le contrôle par modèle interne. Le filtre est du premier ordre défini dans le domaine discret par:

$$v(z) = \frac{1-\omega}{1-\omega z^{-1}}\left[y_{sp}(z) - e(z)\right] \tag{4-6}$$

où ω est le paramètre d'ajustement du filtre et doit être choisi entre 0 et 1. Une valeur près de zéro élimine le filtre et produit une réponse vigoureuse à tout changement de consigne ou à toute perturbation alors qu'une valeur près de un produira une réponse très amortie. La valeur de ω est le choix d'un compromis entre la performance et la robustesse.

Hernandez et coll. [1990] ont intégré un réseau de neurones comme modèle non linéaire dans une structure de contrôle par matrice dynamique (DMC). L'algorithme originel de contrôle par matrice dynamique utilise un modèle linéaire du système et prévoit une correction du modèle à l'aide d'un vecteur d'estimation des perturbations prenant en compte les écarts modèle-système et les perturbations du procédé. L'approche utilisée par Hernandez et coll. [1990] ajoute au vecteur des perturbations les écarts entre les prédictions du modèle linéaire et les prédictions du

modèle non linéaire obtenues par réseau de neurones. Les résultats présentés proviennent de simulations et montrent une nette amélioration pour compenser les perturbations dans le système par rapport à une approche strictement linéaire.

Saint-Donat et coll. [1991] ont implanté un contrôleur prédictif non linéaire multipas (MNPC) dans une structure très similaire à celle de Brengel et coll. [1989] en remplaçant le modèle analytique par un réseau de neurones. C'est une approche multipas appliquée à un système à une variable contrôlée et une variable manipulée, dans laquelle on prend en considération les contraintes inférieures et supérieures sur les variables. Ils ont étudié deux stratégies pour la modélisation prédictive multipas. La première est l'utilisation d'un réseau qui a comme sortie un vecteur de N valeurs futures de la variable contrôlée. Cette approche qui semble attrayante a un défaut majeur ; en effet, le modèle ne peut garantir qu'une valeur prédite pour un temps précis sera la même d'une itération à l'autre. Ces différences sont suffisantes pour empêcher l'algorithme d'éliminer les écarts à la consigne. La deuxième utilise un réseau prédictif à un pas qui est itéré N fois et qui donne de bons résultats.

4.3.3 Contrôleur avec modèle inverse du système

Ydstie [1990] utilise un réseau de neurones comme modèle inverse de la dynamique du système dans une structure de contrôle adaptatif directe (Figure 4.7). Le vecteur d'entrée du réseau de neurones est composé de $\phi(t) = [y(t),y(t-1),\ldots,y(t-n),u(t-2),\ldots,u(t-m+1)]$ et l'erreur de prédiction du réseau est $e(t) = u(t-1) - G_\phi [\phi(t)]$ où G_ϕ est la sortie du réseau après propagation avant. La valeur de contrôle est calculée par $u(t) = G_\phi [\phi(t)^*]$ où $\phi(t)^*$ est égal à $[y(t+1)^*,y(t),y(t-1),\ldots,y(t-n+1),u(t-1),\ldots,u(t-m)]$. La valeur $y(t+1)^*$ est fixée à la valeur de la consigne $y_{sp}(t)$ pour le pas de temps présent.

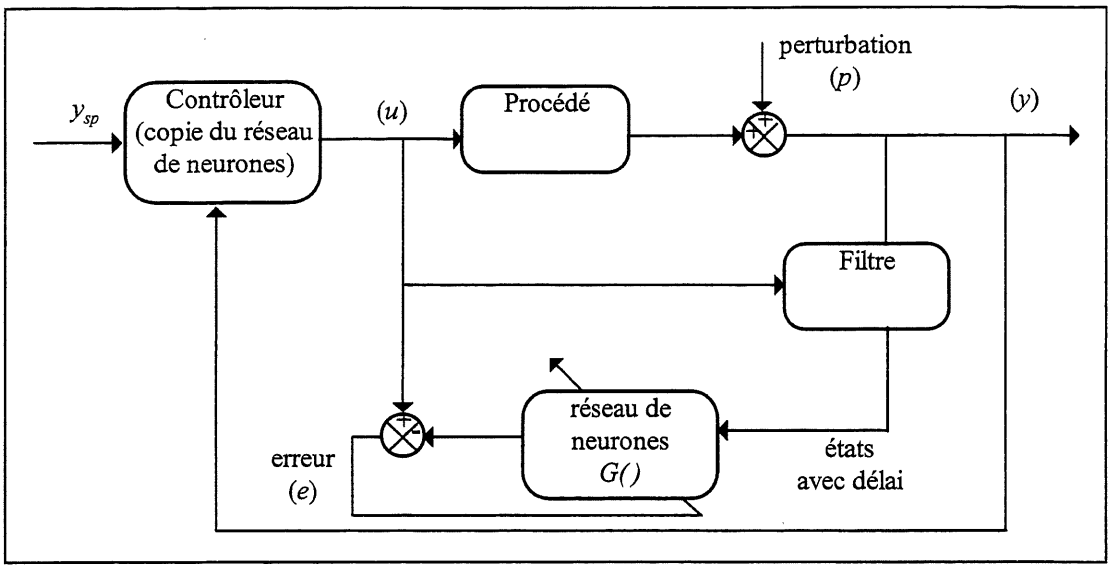


Figure 4.7 Diagramme du contrôleur adaptatif avec modèle inverse direct de Ydstie [1990]

Il n'y a pas de différence entre une approche inverse directe et une approche prédictive à un pas en avant lorsque le système est linéaire et en minimum de phase et que le modèle du système est exact. Mais lorsque l'on est en présence d'un système non linéaire, de perturbation, d'écart modèle-système ou d'un système qui n'est pas en minimum de phase, les lois de contrôle obtenues par les deux approches sont différentes.

Ungar [1990] a utilisé une méthodologie mise au point par Jordan [1988] pour contrôler un bioréacteur. L'approche de Jordan consiste à modéliser la dynamique d'un système à contrôler à l'aide d'un réseau de neurones et d'utiliser ce modèle pour entraîner un autre réseau de neurones qui représente le modèle inverse et qui agit comme contrôleur. Cette approche a été développée pour contrôler des bras manipulateurs, où l'objectif est principalement d'établir une trajectoire d'une façon similaire à celle de Nguyen et Widrow [1990] qui contrôle la trajectoire pour reculer un camion semi-remorque. La particularité de l'approche de Jordan est qu'il incorpore dans la fonction objective du contrôleur à entraîner, des contraintes qui assurent une trajectoire plus continue ou plus douce. Il ajoute aussi des poids différents lorsqu'il y a plusieurs variables contrôlées en fonction de leur importance ou de leur priorité. Les résultats d'Ungar [1990] montrent un écart important entre la consigne et le point d'opération. Cet écart est directement relié à l'écart qui existe entre le modèle et le procédé.

D'une façon générale, les algorithmes développés pour le contrôle des trajectoires où le contrôleur est le résultat d'un apprentissage statique ne sont pas bien adaptés aux procédés en continu du génie chimique.

Bhat et coll. [1990] ont suggéré d'utiliser deux réseaux de neurones dans une structure de contrôle par modèle interne : un réseau pour le modèle du système et un réseau pour le modèle inverse qui est utilisé comme contrôleur. Psichogios et coll. [1991] ont essayé cette approche pour le contrôle du réacteur CSTR non-isotherme étudié par Economou et coll. [1986]. La boucle de contrôle obtenue est incapable de stabiliser les effets d'une perturbation dans le système. La raison de cet échec est attribuée au fait que les deux réseaux de neurones utilisés dans la boucle de contrôle sont entraînés de façon indépendante et que rien ne garantit qu'ils sont l'inverse l'un de l'autre. En fait, dans le cas étudié, une différence de moins de 5% dans la prédiction des deux modèles est suffisante pour empêcher le contrôleur de fonctionner.

Psaltis et coll. [1987] et [1988] ont proposé d'utiliser comme contrôleur un réseau de neurones qui modélise directement l'inverse de la dynamique du système autour du point d'opération et qui est continuellement en entraînement en temps réel. Cette approche est souvent appelée l'apprentissage spécialisé (figure 4.8) par opposition à l'apprentissage généralisé qui modélise, à partir d'un ensemble de données, la dynamique inverse du domaine d'opération.

Pour que y soit égal à y_{sp} à la sortie du procédé, le contrôleur devrait envoyer la commande u^* qui est inconnue. On ne peut donc pas procéder directement à l'entraînement du réseau de neurones par la fonction $Ep = ½(u - u^*)^2$. Pour contourner la difficulté, la fonction erreur qui est minimisée par le réseau de neurones est le carré de la différence entre l'état du système y et la consigne y_{sp} et elle est mesurée à la sortie du procédé. La différence $y - y_{sp}$ est ensuite rétropropagée à travers le modèle du procédé de façon à permettre l'adaptation en continu du contrôleur. Pour un système SISO, on peut écrire une fonction erreur à la sortie du procédé comme:

$$Ep = \frac{1}{2}(y - y_{sp})^2 \tag{4-7}$$

Pour minimiser l a fonction erreur, on doit trouver son gradient par rapport aux poids du réseau de neurones $\frac{\partial Ep}{\partial W_{ij}}$. Le gradient se décompose par la règle d'enchaînement:

$$\frac{\partial Ep}{\partial W_{ij}} = \frac{\partial Ep}{\partial u}\frac{\partial u}{\partial W_{ij}} \tag{4-8}$$

Le terme $\frac{\partial u}{\partial W_{ij}}$ est directement associé au réseau de neurones qui agit comme contrôleur. Pour enclencher la règle « delta généralisé » qui permet de minimiser l'erreur à la sortie du procédé par l'ajustement des poids du réseau de neurones, Psaltis et coll. [1987] ont suggéré de modéliser le procédé et de l'intégrer comme une couche supplémentaire, dont les poids sont fixes, à la fin du réseau de neurones. Par la règle d'enchaînement et en prenant la dérivée de la fonction erreur *Ep* par rapport à la sortie du procédé *y* on trouve que :

$$\frac{\partial Ep}{\partial u} = (y - y_{sp})\frac{\partial y}{\partial u} \tag{4-9}$$

Le terme $\frac{\partial y}{\partial u}$ correspond au jacobien du système, qui est aussi le gain du système au point d'opération. Dans le cas général où il y aurait plusieurs variables mesurées en plus de la variable manipulée, les valeurs du jacobien forment la couche fixe du contrôleur.

Lorsque qu'il n'y a pas de modèle connu du procédé, Psaltis et coll. [1987] déterminent expérimentalement le jacobien local. L'entraînement du contrôleur suit une politique d'identification récursive où, à chaque pas de temps, une itération est effectuée dans la procédure d'optimisation. Les auteurs ont aussi expérimenté une procédure d'entraînement hybride où le contrôleur est d'abord entraîné par un apprentissage généralisé pour cerner le domaine d'opération, avant de l'implanter et d'effectuer l'apprentissage spécialisé. Les résultats obtenus

ne permettent pas d'observer un avantage à procéder à un apprentissage généralisé avant l'apprentissage spécialisé.

Saerens et coll. [1989] ont proposé une variation à l'apprentissage spécialisé de Psaltis et coll. [1987] dans laquelle le modèle du procédé est simplifié pour n'inclure que le signe du jacobien.

$$\frac{\partial Ep}{\partial u} = (y - y_{sp}) \quad signe(\frac{\partial y}{\partial u}) \tag{4-10}$$

Cette simplification est justifiée lorsque le terme $signe(\frac{\partial y}{\partial u})$ est uniquement connu de façon qualitative. En général le gain du procédé est une fonction du point d'opération, mais le signe du gain est constant dans la plage d'opération. Évidemment l'approche de Saerens et coll. [1989] ne peut pas être appliquée pour les procédés dont le gain change de signe comme le réacteur CSTR de Economou et coll. [1986] utilisé dans l'exemple de la section 5.2.

Schiffmann et coll. [1993] ont utilisé l'approche de Saerens et coll. [1989] et l'ont comparée à un contrôleur P.I.D. pour un système linéaire du troisième ordre avec retard. Ils obtiennent avec l'approche par réseau de neurones de meilleurs résultats que le contrôleur P.I.D. réglé à l'aide des règles de Ziegler-Nichols (voir Seborg et coll. [1989]), mais ils constatent qu'en augmentant le paramètre τ_i du contrôleur P.I.D., les réponses des deux méthodes sont pratiquement identiques. Ces résultats ne sont pas surprenants dans la mesure où un contrôleur P.I.D. est performant pour les systèmes linéaires en minimum de phase et qu'il n'y a pas de raison d'obtenir une nette amélioration en utilisant un contrôleur non linéaire, surtout si celui-ci n'est pas dans la classe des contrôleurs prédictifs multipas.

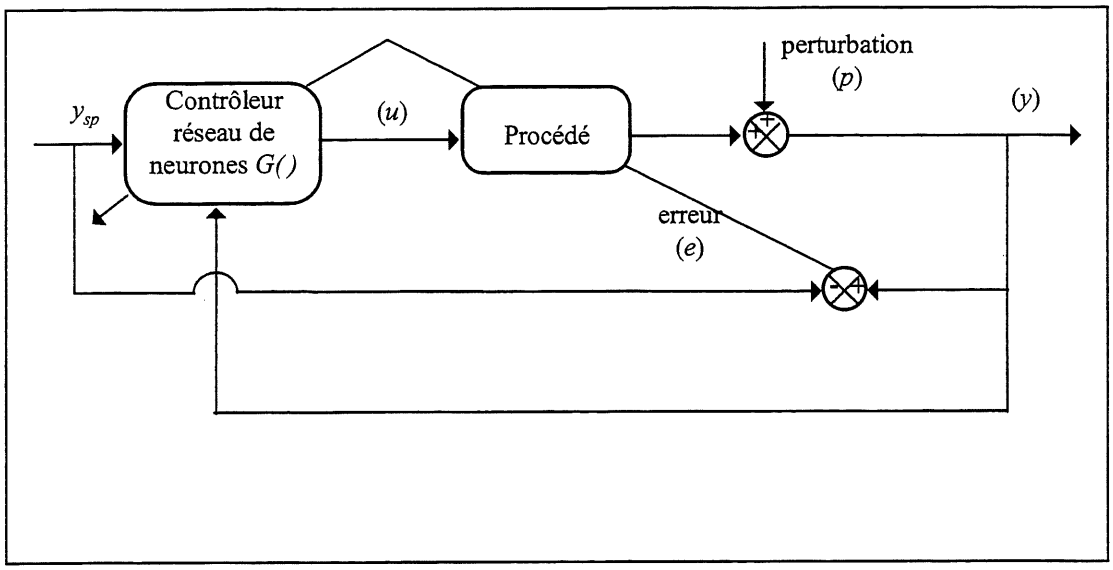


Figure 4.8 Diagramme d'un contrôleur par réseau de neurones entraîné en continu par apprentissage spécialisé.

4.4 **Présentation du contrôleur neuronal adaptatif**

L'approche proposée s'inspire des travaux de Psaltis et coll. [1987] dans le domaine du contrôle à l'aide de réseaux de neurones entraînés par apprentissage spécialisé. Elle se démarque par l'utilisation d'un réseau de neurones distinct pour modéliser le procédé, par une meilleure utilisation des coordonnées de temps et par une nouvelle fonction erreur à minimiser.

Cette approche comporte les avantages des méthodes adaptatives et peut donc s'ajuster à des procédés dont les propriétés dérivent dans le temps. Elle allie aussi les avantages d'utiliser les réseaux de neurones de correspondance pour modéliser la dynamique des procédés, ceux-ci étant reconnus comme de très bons outils d'identification des procédés. L'apprentissage spécialisé offre l'avantage d'adapter le modèle inverse du procédé dans la région d'opération d'intérêt.

4.4.1 Configuration proposée pour un contrôleur neuronal adaptatif

Dans tous les algorithmes de contrôle par entraînement spécialisé, on ne peut évaluer directement l'erreur de la commande u que le contrôleur neuronal adaptatif doit minimiser. On

utilise donc l'erreur entre la consigne et l'état du système qui doit être rétropropagée à travers le procédé afin d'estimer une erreur sur la commande u. Évidemment, l'erreur ne peut être rétropropagée à travers un procédé et l'on doit donc utiliser un modèle sous une forme ou sous une autre. Psaltis et coll. [1987] ont utilisé comme modèle une simple approximation du procédé par son gain alors que Saerens et coll. [1989] ont proposé une variante n'utilisant que le signe du gain. Ces approches, bien que fort simples à implanter, ont un désavantage majeur lors des changements de consigne parce qu'on doit réentraîner le contrôleur dans la nouvelle région uniquement à partir d'une approximation du gain du procédé, c'est-à-dire sans indication précise de la dynamique. Ces approches sont insatisfaisantes pour les procédés où le gain et le temps de réponse varient selon le domaine d'opération, cas que l'on rencontre souvent en génie chimique.

La configuration proposée et schématisée à la figure 4.9 utilise deux réseaux de neurones distincts. Le premier réseau est entraîné comme modèle dynamique du procédé ($\hat{y}(t+1) = F(\,)$), et le second, le contrôleur neuronal adaptatif, est un modèle inverse de la dynamique du procédé ($u(t) = G(\,)$) et il est adapté en continu. L'approche considère l'état actuel y(t), l'état estimé à l'itération précédente $\hat{y}(t)$, l'état estimé actuel $\hat{y}(t+1)$ du procédé et la valeur de consigne $y_{sp}(t)$ et rétropropage l'erreur E(t) à travers le modèle du procédé puis à travers le contrôleur adaptatif pour déterminer la direction dans laquelle doit se prendre l'action. Une fois la direction déterminée, la meilleure commande $u(t)$ est évaluée en estimant son impact à l'aide du modèle prédictif. Par la suite le contrôleur est adapté.

Il y a plusieurs avantages à utiliser deux réseaux de neurones distincts. Un modèle fixe du procédé représente la dynamique dans l'ensemble du domaine d'opération du procédé et il est remis à jour uniquement lorsque la dynamique a évolué et que l'on dispose d'une bonne base de données pour en faire l'adaptation. Le contrôleur neuronal adaptatif est adapté en permanence dans le domaine d'intérêt et peut suivre les procédés ayant des caractéristiques qui dérivent dans le temps. L'utilisation d'un modèle dynamique du procédé dans une structure de contrôle par apprentissage spécialisé permet d'avoir une composante prédictive dans la fonction erreur qui est minimisée. De plus, ce modèle permet de fournir en tout temps la dynamique du procédé et d'estimer rapidement la direction dans laquelle le contrôleur neuronal adaptatif doit évoluer.

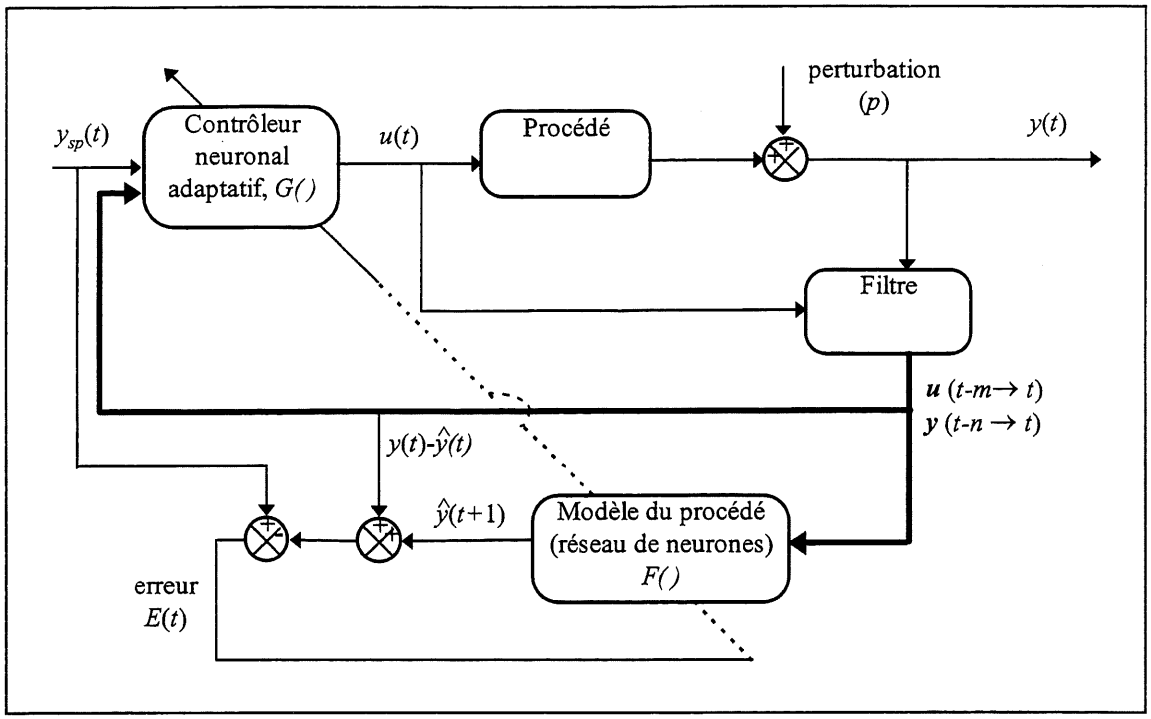


Figure 4.9 Configuration du contrôleur neuronal adaptatif.

4.4.2 Algorithme de contrôle

On peut considérer que le système, tel qu'il est présenté à la figure 4.10, est fait de deux parties. La première est le contrôleur neuronal adaptatif entraîné pour trouver la commande $u_p(t+d)$ (la constante d représente le retard pur dans le procédé). La seconde partie est le modèle dynamique par réseau de neurones préalablement entraîné. Les deux réseaux sont connectés par une fonction linéaire Lu qui fait la mise à l'échelle de u_p pour obtenir la commande u. Les deux réseaux ont une structure très similaire avec le même nombre d'éléments dans le vecteur d'entrée. En fait seuls deux éléments sont intervertis : le contrôleur neuronal a la consigne $y_{sp}(t)$ à l'entrée et la commande $u_p(t+d)$ à la sortie, tandis que le modèle a la commande $u(t+d)$ à l'entrée et la valeur estimée $\hat{y}(t+1)$ à la sortie. Le nombre de neurones dans la couche cachée peut être différent pour les deux réseaux. Pour simplifier la notation, la constante de retard d est éliminée pour la suite du chapitre.

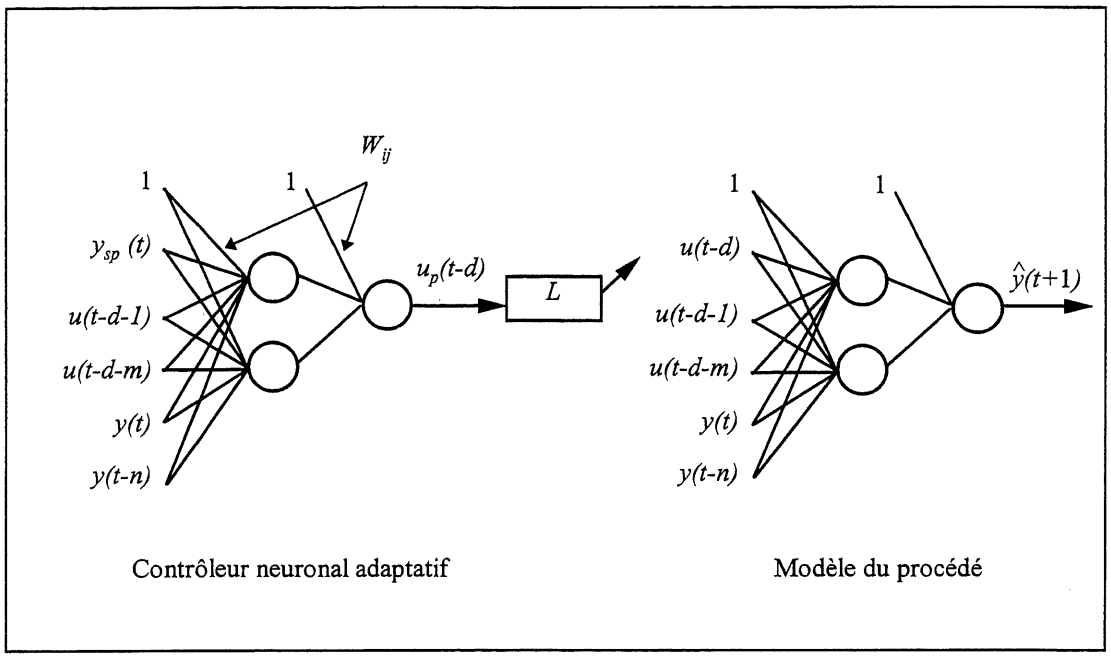


Figure 4.10 Schéma des réseaux de neurones utilisés pour le contrôleur neuronal adaptatif et le modèle du procédé. Les deux réseaux sont reliés par une fonction linéaire L de mise à l'échelle entre u_p et u.

L'entraînement du contrôleur suit une politique d'identification récursive où, à chaque pas de temps, une itération est effectuée dans la procédure d'optimisation. Nous voulons donc estimer la direction dans laquelle *u(t)* doit bouger pour minimiser la différence entre la variable contrôlée à la sortie du procédé et la consigne. Pour ce faire, il faut évaluer le gradient entre l'erreur à la sortie du procédé et les poids du réseau de neurones du contrôleur. Cela se fait très simplement si l'on considère les deux réseaux comme un seul réseau comportant plusieurs couches cachées et où l'erreur évaluée à la sortie du modèle est rétropropagée à chaque couche intermédiaire par la règle du delta généralisé. De cette façon, la valeur de l'erreur de la commande n'a pas à être évaluée directement.

Dans l'algorithme qu'ils ont présenté, Psaltis et coll. [1987] utilisent l'erreur entre l'état actuel du procédé et la consigne pour évaluer la commande pour la prochaine itération. Cette approche entraîne des incohérences dans l'utilisation des cordonnées de temps, car on utilise alors une estimation de l'erreur ($u_p(t\text{-}1)$ - $u^*(t\text{-}1)$) pour calculer le gradient, modifier les poids, minimiser l'erreur et évaluer ensuite la commande $u_p(t)$. Un des avantages à l'utilisation d'un

modèle dynamique du procédé est qu'il permet d'implanter une structure prédictive à un pas en calculant une erreur qui est fonction de la valeur estimée pour la prochaine itération et de la consigne couvrant cette période. De cette façon, on n'utilise donc pas l'erreur au temps présent pour adapter le contrôleur pour la prochaine commande mais bien l'erreur estimée pour la prochaine itération. Cette distinction est essentielle pour l'algorithme. Cette façon de faire permet d'évaluer une direction de descente qui minimise les poids du contrôleur neuronal et qui pourra être utilisée en conjonction avec un algorithme de recherche unidirectionnelle en estimant l'impact d'une variation de la commande sur la valeur estimée $\hat{y}(t+1)$ produite par le modèle. Ceci est un net avantage puisqu'il permet d'utiliser tous les algorithmes d'optimisation énumérés au Chapitre 2 pour faire l'entraînement du contrôleur car, à l'exemption de rétro-propagation, ils ont tous recourt à une méthode de recherche unidirectionnelle. L'algorithme de rétro-propagation, utilisé par Psaltis et coll. [1987] et Saerens et coll. [1989] est particulièrement à éviter car, avec un pas de descente fixe ,cet algorithme ne minimise nullement de façon optimale l'erreur à chaque itération.

Pour prendre en compte les différences entre le modèle et le procédé et minimiser les écarts par rapport à la consigne, on doit aussi inclure dans la fonction erreur l'état actuel du procédé. La fonction erreur utilisée est définie par:

$$E(t) = \tfrac{1}{2}\left\{ y_{sp}(t) - \left(\hat{y}(t+1) + \left[y(t) - \hat{y}(t) \right] \right) \right\}^2 \qquad (4\text{-}11)$$

Cette fonction est donc la différence au carré entre la consigne et la valeur estimée par le modèle, valeur qui est augmentée de la différence entre l'état actuel du procédé et l'estimation pour la présente itération. On peut appeler cette fonction une erreur projetée corrigée. La minimisation de cette fonction ne garantit pas qu'il n'y aura aucun écart par rapport à la consigne, mais cet écart devrait toujours être très faible. Dans une région d'opération précise, un modèle dynamique par réseau de neurones a tendance à avoir un écart constant par rapport au procédé. Le système contrôlé par la minimisation de l'équation (4-11) tendra donc vers la consigne.

L'algorithme de contrôle à l'aide du contrôleur neuronal adaptatif est constitué des étapes suivantes:

- Déterminer la consigne pour la prochaine itération, $y_{SP}(t)$.
- Faire une propagation avant de l'entrée du contrôleur neuronal adaptatif jusqu'à la sortie du modèle du procédé.
- Évaluer l'erreur $E(t) = \frac{1}{2}\left\{ y_{sp}(t) - \left(\hat{y}(t+1) + \left[y(t) - \hat{y}(t) \right] \right) \right\}^2$.
- Rétropropager l'erreur par la règle du delta généralisé jusqu'aux poids du contrôleur et évaluer le gradient de l'erreur par rapport aux poids.
- Déterminer une direction de descente en fonction de l'algorithme d'optimisation retenu. Cette direction de descente est la loi d'adaptation du contrôleur.
- Effectuer une recherche unidirectionnelle dans la direction de descente par une propagation à travers le modèle du procédé et adapter les poids du contrôleur.
- Évaluer la nouvelle commande $u(t)$.

4.4.3 Loi d'adaptation du contrôleur

Appelons $\boldsymbol{G}$ le vecteur du gradient de l'erreur E(t) par rapport aux poids. Comme le contrôleur suit une politique d'identification récursive où, à chaque pas de temps, une itération est effectuée dans la procédure d'optimisation, $\boldsymbol{G}(t)$ est égal à $\boldsymbol{G}(n)$. Une direction de descente $\boldsymbol{S}(t)$ est calculé à partir du gradient $\boldsymbol{G}(t)$. Les poids du contrôleur sont adaptés selon cette direction de descente à l'aide d'un algorithme de recherche unidirectionnelle qui est décrit à la section suivante (4.4.4).

Choisissons une fonction linéaire L de mise à l'échelle entre u et u_p, de façon à ce que les gradients $\frac{\partial E}{\partial u_p(t)}$ et $\frac{\partial E}{\partial u(t)}$ soient proportionnels.

$$\frac{\partial E}{\partial u_p(t)} = \sigma \frac{\partial E}{\partial u(t)} \tag{4-12}$$

La valeur de la constante σ de proportionnalité n'a pas d'effet sur le calcul de la direction de descente.

Le gradient $\frac{\partial E}{\partial u(t)}$ est évalué entre l'entrée et la sortie du modèle dynamique (réseau de droite sur la figure 4.10). Il peut facilement être calculé par la règle du delta généralisé décrite à la section 2.4. Pour ce réseau de neurones, notons j le nombre de neurones dans la couche cachée. Pour ce cas, seul l'impact de l'erreur sur la commande en position un du vecteur d'entrée est calculé. Reprenant la notation du chapitre 1, i est égal à 1 et, puisqu'il n'y a qu'un seul neurone de sortie, k est égal à 1. Par analogie avec l'équation (2-23) on peut poser :

$$\frac{\partial E}{\partial u(t)} = -\sum_j \delta_j W_{ij} \tag{4-13}$$

et par l'équation (2-24)

$$\delta_j = f'_j(sum_j)\,\delta_k\,W_{jk} \tag{4-14}$$

Le terme δ_k est évalué de la même façon que dans l'équation (2-21), mais en utilisant la fonction objective (4-11) ; il s'écrit :

$$\delta_k = f'_k(sum_k)\left\{y_{sp} - \left[\hat{y}(t+1) + (y(t) - \hat{y}(t))\right]\right\} \tag{4-15}$$

À l'aide des équations (4-13), (4-14) et (4-15) le gradient se réécrit comme :

$$\frac{\partial E}{\partial u(t)} = -\sum_j f'_j(sum_j)\,\delta_k\,f'_k(sum_k)\left\{y_{sp} - \left[\hat{y}(t+1) + (y(t) - \hat{y}(t))\right]\right\} W_{jk}\,W_{ij} \tag{4-16}$$

avec i et k sont égaux à 1 tel que vu précédemment.

Pour calculer le gradient de l'erreur par rapport aux poids du réseau du contrôleur neuronal, on reprend les équations (2-34) et (2-35) en remplaçant $-\left(Y_j - O_j\right)$ par $\frac{\partial E}{\partial u(t)}$. Donc le gradient s'évalue pour la couche finale par :

$$\frac{\partial E}{\partial W_{jk}^{(2)}} = f'^{(2)}(sum_k).\frac{\partial E}{\partial u(t)}.H_j \tag{4-17}$$

et pour la couche intermédiaire par :

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = f'^{(1)}(sum_j).\sum_{k=1}^{o}\left[f'^{(2)}(sum_k).\frac{\partial E}{\partial u(t)}.W_{jk}^{(2)} \right].X_i \tag{4-18}$$

Le gradient $\boldsymbol{G}(t)$ est constitué de tous les éléments des équations (4-17) et (4-18). Dans les exemples du chapitre 5, l'algorithme BFGS-N est utilisé pour calculer une direction de descente $\boldsymbol{S}(t)$ qui incorpore les éléments des interactions du second ordre entre les poids arrivant à un même neurone du contrôleur neuronal. Cet algorithme s'est montré très efficace dans toutes les situations d'adaptation en continu. Lors de changements de consigne importants, les matrices hessiennes approximées peuvent devenir non-définies positives et sont remplacées par des matrices identités. Dans ces conditions, l'algorithme se comporte comme une méthode de gradient simple. Dans les autres situations, lorsque le contrôleur fait de la régulation ou lorsque le système s'approche du point de consigne, on peut évaluer les interactions du second ordre et obtenir ainsi une meilleure direction de descente.

Le problème des minimums locaux dans l'optimisation des réseaux de neurones est bien connu et bien documenté et l'on pourrait craindre que l'adaptation en continu du contrôleur souffre de ce problème. Mais comme le vecteur d'entrée du réseau contrôleur est toujours en changement, il y a très peu de chance de se trouver dans un minimum local puisque le domaine dans lequel se fait l'optimisation varie continuellement. En fait, et bien que cela ne constitue pas une démonstration, aucun essai effectué avec le contrôleur neuronal adaptatif n'a conduit à un minimum local.

L'initialisation des poids du contrôleur se fait avec des valeurs aléatoires. Dans des applications critiques, une copie du modèle dynamique peut être utilisée pour obtenir un ensemble de poids de départ qui produit une commande initiale dans la plage d'opération. Généralement peu d'itérations sont nécessaires pour initialiser le contrôleur (typiquement de 30 à 40) mais cela dépend de la qualité du modèle dynamique du système.

4.4.4 Algorithme de recherche unidirectionnelle

L'objectif de l'algorithme de recherche unidirectionnelle est de trouver le scalaire λ qui minimise la fonction erreur estimée par réseaux de neurones (équation (4-11)) à la sortie du modèle dynamique. Les poids du contrôleur neuronal adaptatif sont ajustés à l'aide de cette quantité le long de la direction de descente, selon l'équation (2-49) :

$$\Delta \boldsymbol{W}(t) = \lambda(t)\boldsymbol{S}(t) \tag{4-19}$$

De plus, dans l'algorithme, deux contraintes sont ajoutées au contrôleur. La première contrainte assure que la commande du contrôleur reste à l'intérieur de bornes inférieure et supérieure,

$$u_{min} < u < u_{max} \tag{4-20}$$

Les valeurs de ces bornes sont souvent fixées par la nature de la variable manipulée. On doit ajuster la fonction L de telle sorte que, avec l'utilisation d'une fonction d'activation sigmoïde, lorsque u_p est égale à 0.1 la commande est à sa valeur inférieure u_{min} et lorsque u_p est égale à 0.9 la commande est à sa valeur supérieure u_{max}. Cette précaution est nécessaire afin d'éviter que les poids du contrôleur neuronal adaptatif deviennent très grands en essayant de générer une valeur u_p qui tend vers les bornes de la fonction d'activation, ce qui pourrait entraîner des difficultés à adapter le contrôleur lors d'un changement de consigne.

La seconde contrainte impose une limite sur la vélocité de la commande entre deux temps d'échantillonnage. Cette limite est en fait le seul paramètre d'ajustement du contrôleur.

$$\left|u_p(t-1)-u_p(t)\right| \leq \Theta \qquad (4\text{-}21)$$

La valeur de Θ est déterminée par l'utilisateur et reflète la confiance dans le modèle dynamique du procédé ainsi que le degré de robustesse désiré dans le système de contrôle.

L'algorithme est composé des étapes suivantes:

1) Poser λ=1, le plein pas de Newton que nous voulons utiliser le plus souvent possible.
2) Calculer des nouveaux poids par l'équation (4-19) et la commande $u_p(t)$.
3) Évaluer $E(t)$ en propageant $u(t)$ dans le modèle par réseau de neurones.
4) Vérifier que $E(t) < E(t\text{-}1)$ et que les contraintes, équations (4-20) et (4-21), sont respectées. Si tel est le cas, fin de l'algorithme. Sinon, passer au point 5.
5) Poser que $\lambda = \lambda/2$ et retour au point 2. On vérifie aussi si $\lambda < \lambda_{min}$ auquel cas la recherche est terminée.

La valeur de λ_{min} est généralement très faible ($\cong 10^{-5}$) de sorte que, lorsqu'elle est atteinte, $u_p(t) \cong u_p(t\text{-}1)$. Ce cas se produit principalement lorsque le contrôleur atteint une des bornes de l'équation (4.20) ou, lorsque suite à une perturbation importante, la différence $\left[y(t)-\hat{y}(t)\right]$ est devenue telle dans l'équation (4-11) que la fonction erreur ne pourra pas être minimisée pour cette itération.

5. EXEMPLES DU CONTRÔLEUR NEURONAL ADAPTATIF

5.1 Introduction

Ce chapitre développe deux exemples d'utilisation du contrôleur neuronal adaptatif. Le premier exemple traite du contrôle d'un bioréacteur CFSTR ayant une dynamique complexe avec différents états d'équilibre et un point de bifurcation. Dans l'autre exemple, un réacteur CSTR non-isotherme avec une réaction réversible du premier ordre sert de banc d'essai. Chaque exemple présente les états d'équilibre du système ainsi que les valeurs et les résultats de l'entraînement du modèle du système par un réseau de neurones. Dans ce chapitre on entend par procédé le modèle idéal provenant des équations d'état.

5.2 Contrôle d'un bioréacteur

Le contrôle de procédés qui comportent des états d'équilibre multiples, avec des points de bifurcation vers des solutions oscillantes en régime permanent et des cycles limites, constitue un défi important pour tout algorithme de contrôle. Ces phénomènes oscillatoires sont bien connus et ont été observés pour certains réacteurs CSTR lors de réactions exothermiques. Des phénomènes similaires ont aussi été observés lors de la culture de cellules dans des bioréacteurs.

Agrawal et coll. [1982] rapportent les travaux expérimentaux de cinq équipes qui ont observé des phénomènes oscillatoires lors de la culture d'une seule espèce de cellules dans un bioréacteur en continu. Cela les a amenés à étudier le comportement théorique de la dynamique d'un bioréacteur isotherme en continu et plus particulièrement les modèles pouvant conduire à des systèmes présentant des caractéristiques oscillantes. Un des modèles qu'ils ont étudiés et qui utilise une cinétique inhibée par la présence de substrat, est devenu un banc d'essai pour divers algorithmes de contrôle. Citons les travaux de Brengel et coll. [1989] qui ont contrôlé le niveau de biomasse et de substrat de ce système avec leur algorithme "Multistep Nonlinear Predictive Controller". Les auteurs obtiennent des résultats intéressants dans la mesure où ils ont une bonne connaissance des deux paramètres importants du système que sont le taux de réaction et le coefficient de rendement. Ils contrôlent le niveau de substrat (S) et de biomasse (X) autour d'un

point d'équilibre oscillant en manipulant le débit d'alimentation. Dans leur exemple, le substrat et la biomasse sont mesurés à un intervalle de temps suffisamment rapide par rapport à la dynamique du système pour obtenir en tout temps un modèle très précis. Comme ils le précisent, ce système est sous-contrôlé avec deux variables contrôlées pour une seule variable manipulée.

Ungar [1990] suggère d'utiliser ce même système comme banc d'essai pour le contrôle de procédés par des algorithmes à base de réseaux de neurones. Les résultats qu'il présente utilisent la méthodologie mise au point par Jordan [1988] pour le contrôle des robots. Avec cette approche, la qualité de la réponse obtenue est directement fonction de la précision du modèle de la dynamique du système obtenue par le réseau de neurones. Avec un petit réseau de deux couches cachées composées de cinq neurones chacune, il existe un écart important entre la consigne et la valeur d'équilibre du système.

5.2.1 Modèle dynamique du bioréacteur étudié

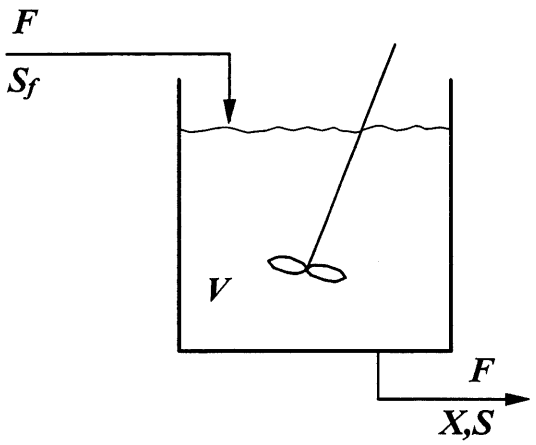
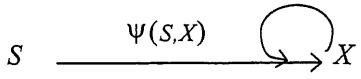


Figure 5.1 Schéma du bioréacteur bien mélangé en continu

Le système schématisé à la figure 5.1 consiste en une réaction autocatalytique de type



(5-1)

où le taux de réaction $\psi(S,X) = \mu(S)\,X$.

Dans ce système, le taux spécifique de croissance $\mu(S)$ est inhibé par la présence de substrat en forte concentration et est décrit par le modèle à deux paramètres « one-hump ».

$$\mu(S) = k_c\, S\, e^{-S/K} \tag{5-2}$$

Le taux spécifique de croissance de ce modèle atteint son maximum lorsque S est égale à K. Agrawal et coll. [1982] ont utilisé ce modèle à deux paramètres à la place des modèles plus populaires à trois ou quatre paramètres, comme le modèle de Haldane, afin de minimiser l'étude paramétrique du système. Le taux spécifique de consommation de substrat $\sigma(S)$ est relié au taux spécifique de croissance par un coefficient de rendement Y qui n'est pas constant:

$$Y(S) = \frac{\mu(S)}{\sigma(S)} = a + bS \tag{5-3}$$

Agrawal et coll. [1982] ont démontré que la présence d'un point de bifurcation est impossible lorsque le coefficient de rendement (Y) est constant. Ils ont choisi un coefficient de rendement qui est une simple fonction linéaire du substrat dans le but de minimiser l'étude paramétrique du système.

Pour un bioréacteur CSTR, les bilans de masses sur la biomasse et le substrat donnent:

$$\frac{dX}{dt} = -DX + \mu(S)\,X \tag{5-4}$$

$$\frac{dS}{dt} = D(S_f - S) - \sigma(S)\,X \tag{5-5}$$

où D est le taux de dilution égal au rapport du débit massique F sur le volume du réacteur V. Le réacteur pourra présenter des états de régime cyclique pour certaines valeurs des

constantes a et b du coefficient de rendement et pour certaines valeurs de K du modèle cinétique. Agrawal et coll. [1982] ont regroupé ces constantes dans deux nombres adimensionnels, β et γ :

$$\beta = \frac{a}{b\, S_f} \tag{5-6}$$

$$\gamma = \frac{K}{S_f} \tag{5-7}$$

Ils ont énuméré les conditions nécessaires pour être en présence d'un point de bifurcation. Brengel et coll. [1989] ont utilisé dans leur étude, des valeurs de β=0.02 et γ=0.48 qui ont été par la suite reprises par Ungar[1990]. Les bilans de masse sont réécrits sous une forme adimensionnelle en fonction des variables C_1, représentant la masse adimensionnelle des cellules, et C_2, la conversion de substrat.

$$C_1 = \frac{X}{S_f\, Y(S_f)} \tag{5-8}$$

$$C_2 = \frac{(S_f - S)}{S_f} \tag{5-9}$$

Da, le nombre de Damkohler, est égal au rapport du taux spécifique de croissance aux concentrations d'entrée sur le taux de dilution dans le réacteur ; τ est le temps de séjour adimensionnel.

$$Da = \frac{\mu(S_f)}{D} \tag{5-10}$$

$$\tau = t\, D \tag{5-11}$$

Les équations des bilans de masse deviennent :

$$\frac{dC_1}{d\tau} = -C_1 + Da\ C_1\ (1-C_2)\ e^{C_2/\gamma} \tag{5-12}$$

$$\frac{dC_2}{d\tau} = -C_2 + Da\ C_1\ \frac{1+\beta}{1+\beta-C_2}(1-C_2)\ e^{C_2/\gamma} \tag{5-13}$$

5.2.2 Analyse de l'état d'équilibre du système

Cette section présente une analyse des états d'équilibre du système. Le système est étudié pour les valeurs de β=0.02 et γ=0.48. A l'équilibre, les dérivées $\frac{dC_1}{d\tau}$ et $\frac{dC_2}{d\tau}$ sont nulles et l'équation (5-12) conduit à :

$$Da\ (1-C_2)\ e^{C_2/\gamma} = 1 \tag{5-14}$$

et l'équation (5-13) conduit à :

$$C_2 = \frac{1+\beta}{1+\beta-C_2}\ C_1 \tag{5-15}$$

Les conditions de lessivage du système conduisent à C_1 = 0. Par l'équation (5-15), on trouve qu'en condition de lessivage C_2 est aussi égal à 0 et dans ce cas l'équation (5-14) implique que *Da* est égal à 1.

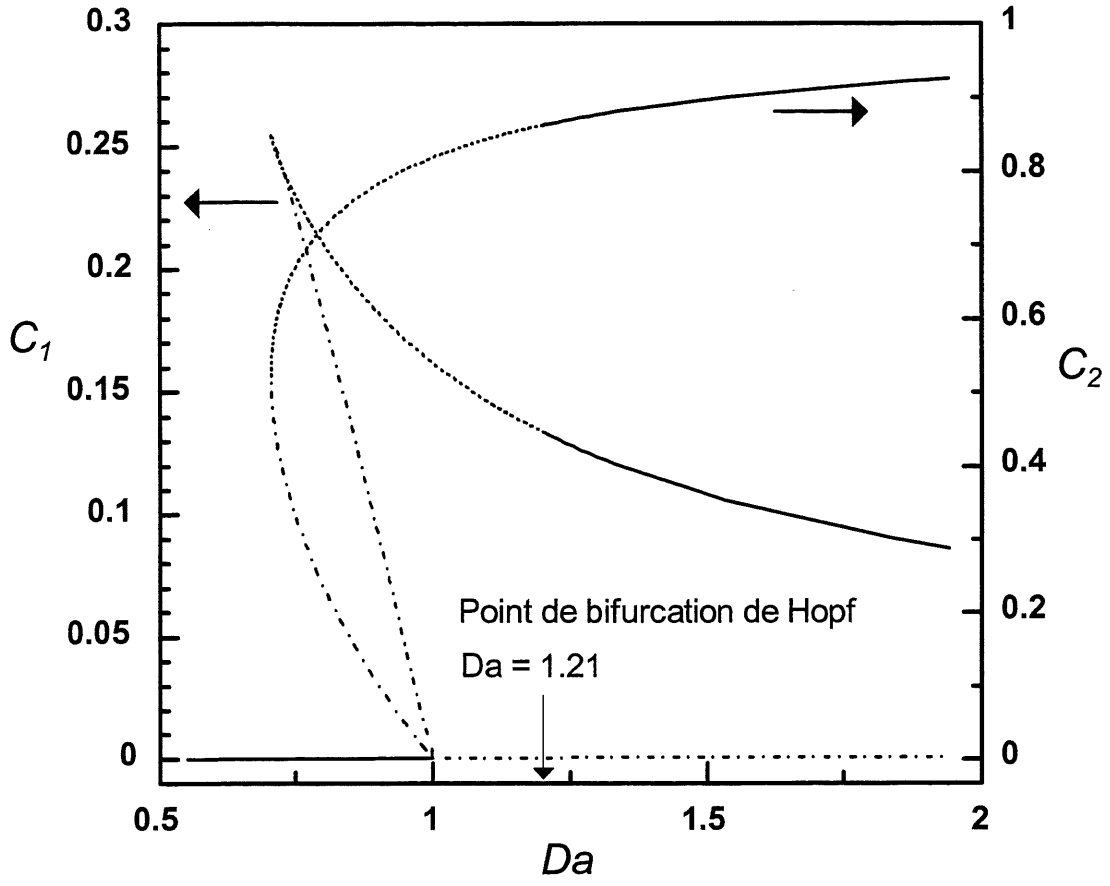


Figure 5.2 Graphique des états d'équilibres du bioréacteur lorsque β=0.02 et γ=0.48. Les trois régimes représentés sont:

———— : stable

- - - - - - - - - - : oscillation périodique

— - — - — - — : instable

La figure 5.2 présente les états d'équilibre de ce système calculés à partir des équations (5-14) et (5-15). Le point de bifurcation de Hopf caractérise le passage d'un état d'équilibre stable à une solution périodique ; il apparaît lorsque les valeurs propres du jacobien du système deviennent purement imaginaires et il est localisé autour de $Da = 1.21$. Au point maximal de production de cellules, lorsque Da est égal à 0.705, le jacobien du système est singulier. A ce moment là C_1 est égal à 0.255 et C_2 à 0.52. Ce point est associé au moment où le taux spécifique de croissance est maximal et qui se produit lorsque S est égal à K.

On peut remarquer que le système est particulièrement complexe lorsque le nombre de Damkohler est inférieur aux conditions de lessivage ($Da < 1$) et supérieur à la valeur qui correspond au taux spécifique de croissance maximal ($Da > 0.705$). Dans cette région, il y a trois états d'équilibre possible pour chaque valeur du nombre de Damkohler : un état stable qui correspond aux conditions de lessivage ; un état instable et un état d'oscillation périodique.

5.2.3 Modélisation du bioréacteur par réseaux de neurones

La représentation de la dynamique du procédé par un réseau de neurones est une composante essentielle du contrôleur neuronal adaptatif. La structure de l'entrée du réseau est une fonction des variables mesurées disponibles. Dans cet exemple, l'on considérera que les concentrations de cellule et de substrat sont mesurées à une période $\Delta\tau = 0.5$. Le modèle du bioréacteur est en non-minimum de phase avec un temps d'inversion qui peut atteindre $\tau = 0.4$ dans la zone d'équilibre stable. La période d'échantillonnage choisie est donc suffisamment longue pour permettre au système de s'inverser, comme cela est requis pour les systèmes de contrôle prédictif à un pas.

La couche d'entrée du réseau est composée de 3 valeurs, $Da(n)$, $C_1(n)$ et $C_2(n)$ qui sont respectivement la variable manipulée (taux de dilution normalisé), la variable contrôlée (concentration normalisée de la biomasse) et une variable mesurée (concentration normalisée du substrat) du système. La couche de sortie a un seul neurone qui représente la variable contrôlée $C_1(n+1)$ à un pas de temps en avant $\Delta\tau$. La variable contrôlée est normalisée entre 0 et 1 lorsqu'une fonction d'activation sigmoïde est utilisée. Une couche cachée constituée de 15 neurones procure assez de degrés de liberté pour bien capter la dynamique du procédé.

Le choix du point d'opération d'un procédé est basé à partir de considérations économiques. En général, on opérera un réacteur à un point où le rendement sera maximal tout en permettant une production fiable. Pour ce bioréacteur, le point d'opération le plus intéressant se situe à l'intérieur de la zone d'équilibre stable près du point de bifurcation. A cet endroit, il est possible d'obtenir un bon rendement et un contrôle robuste. Au-delà du point de bifurcation, dans

la zone d'oscillation périodique, le contrôle est très difficile et entraîne une manipulation excessive de la variable *Da*. De plus, dans cette zone, le temps d'inversion du système peut-être supérieur à la période d'échantillonnage de $\Delta\tau = 0.5$. On devrait donc utiliser soit une méthode prédictive à plusieurs pas, soit une période d'échantillonnage plus longue .

La figure 5.3 reprend les états d'équilibre présentés à la figure 5.2 en y ajoutant le domaine d'opération dynamique modélisé. En abscisse, le nombre de Damkohler varie de 1.2 à 2.0 . Pour C_1, la zone couverte s'étend approximativement de 0.08 à 0.155, alors que la variable C_2 associée varie de 0.80 à 0.95.

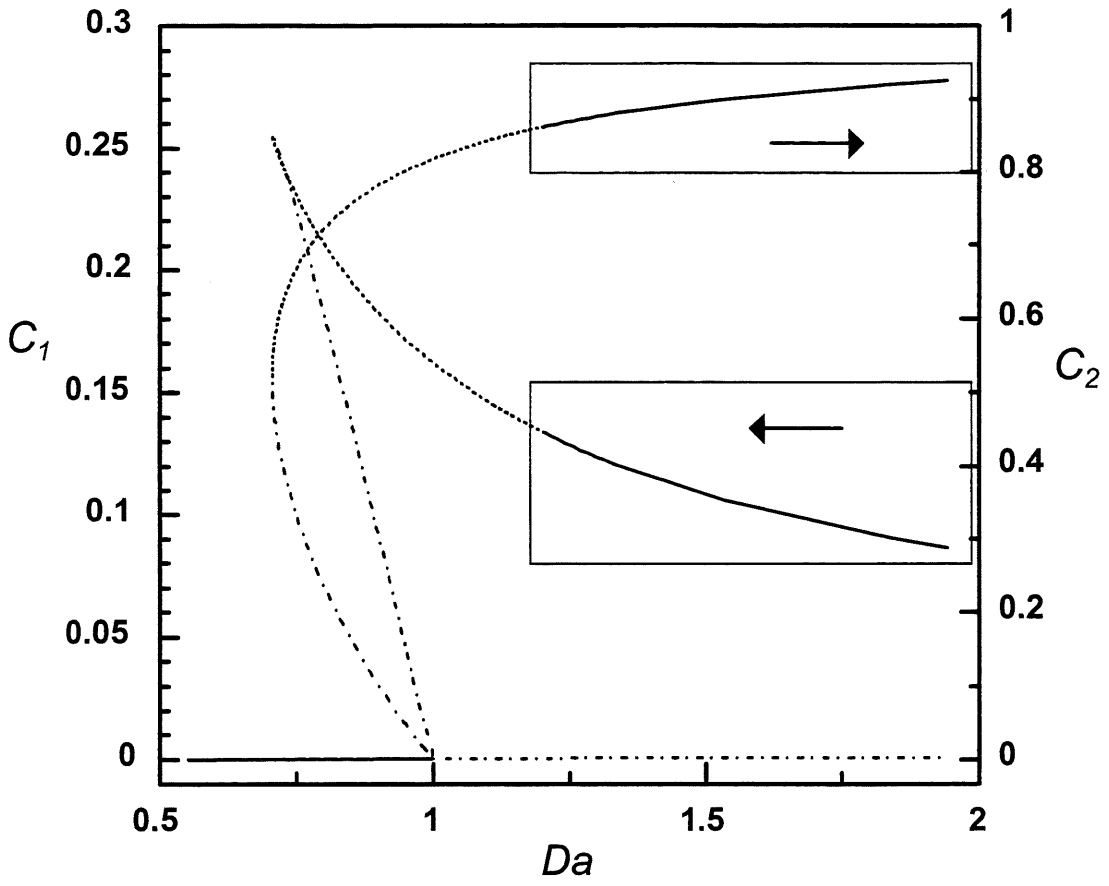


Figure 5.3 Graphique des états d'équilibre du bioréacteur. Les plages entourées représentent le domaine d'opération dynamique modélisé. Les trois régimes représentés sont:

——————— : stable

- - - - - - - - - - - - : oscillation périodique

— - — - — - — : stable

Les valeurs de $C_1(n)$, $C_2(n)$, $Da(n)$ et $C_1(n+1)$ utilisées pour entraîner le réseau de neurones et pour valider l'entraînement sont générées en intégrant les équations de bilans de masse (5-12) et (5-13) à partir d'un signal Da variant d'un point initial égal à 1.6 et des conditions d'équilibre correspondantes $C_1 = 0.102$ et $C_2 = 0.905$. L'intégration des systèmes d'équations différentielles qui comportent un point de bifurcation et des cycles limites, est délicate et une attention particulière doit être accordée au choix de la méthode numérique. En règle générale, les méthodes explicites sont à proscrire pour ces systèmes pour des raisons de stabilité numérique. Dans cet exemple, la méthode implicite du trapèze solutionnée par Newton-

Raphson est utilisée. Le signal provient d'une suite de valeur *Da* qui varie avec une composante aléatoire. L'équation qui génère le signal a la forme suivante:

$$Da(n+1) = Da(n) + \sigma \cdot 1.6 \qquad (5\text{-}16)$$

où σ est compris entre 0.1 et -0.1.

La variation de *Da* d'une itération à l'autre est bornée à un maximum de 10% de la valeur moyenne de la plage d'opération (*Da* = 1.6), ce qui permet de bien se concentrer sur la plage dynamique dans laquelle va évoluer le système. Lorsque le contrôleur neuronal adaptatif utilise le modèle généré à partir de ces valeurs, le changement maximum entre deux instants d'échantillonage (le paramètre Θ) ne devrait pas excéder 10%. Pour s'assurer de bien couvrir le domaine dynamique, le signal *Da* utilisé pour générer les valeurs d'entraînement est constitué de deux séries de 120 données (Figure 5.4) (*P* = 240). La première série couvre la zone de *Da* comprise entre 1.2 et 1.6 et la seconde couvre la zone de *Da* entre 1.6 et 2.0. De la même façon, le signal utilisé pour générer les valeurs de validation provient de deux séries de 50 données (Figure 5.6) (*P*= 100) qui couvrent respectivement les deux zones. La figure 5.5 présente les valeurs de C_1 et C_2 obtenues après intégration des équations de bilan de masse pour le signal d'entraînement de la figure 5.4, et de la même façon, la figure 5.7 correspond aux valeurs utilisées pour la validation.

L'analyse de ces graphiques montre que lorsque les valeurs de *Da* sont plus faibles les valeurs de C_1 sont plus élevées. Et de plus les valeurs de C_1 et de C_2 oscillent beaucoup. C'est lorsque *Da* approche du point de bifurcation (≅1.2) que la dynamique du système est la plus complexe.

Les graphiques des figures 5.5 et 5.7 présentent aussi le résultat de l'apprentissage du réseau de neurones obtenu après 400 itérations. Après 400 itérations, un plateau est atteint dans la minimisation de la fonction d'erreur des valeurs de validation, ce qui indique qu'il serait inutile de poursuivre l'entraînement. À ce stade, la valeur de la fonction erreur *E* définie au chapitre 1 (équation 2.6), divisée par le nombre de valeurs *P* est de 1.2×10^{-8} pour l'entraînement et de

$2.2x10^{-8}$ pour la validation. La ligne continue représente les valeurs cibles et le cercle représente les valeurs obtenues par le réseau de neurones. Les résultats du réseau de neurones sont excellents tant pour les valeurs d'entraînement que pour les valeurs de validation. Les seuls petits écarts se trouvent lorsque la dynamique est la plus complexe, c'est à dire lorsque les valeurs de *Da* sont les plus faibles.

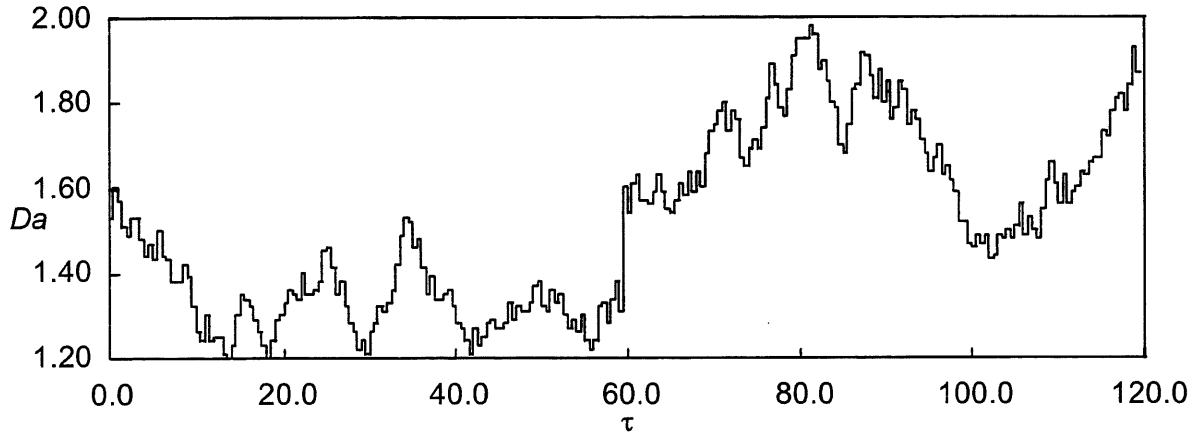


Figure 5.4 Signal de la variable manipulée *Da* utilisée pour générer les valeurs d'entraînements des variables C_1 et C_2.

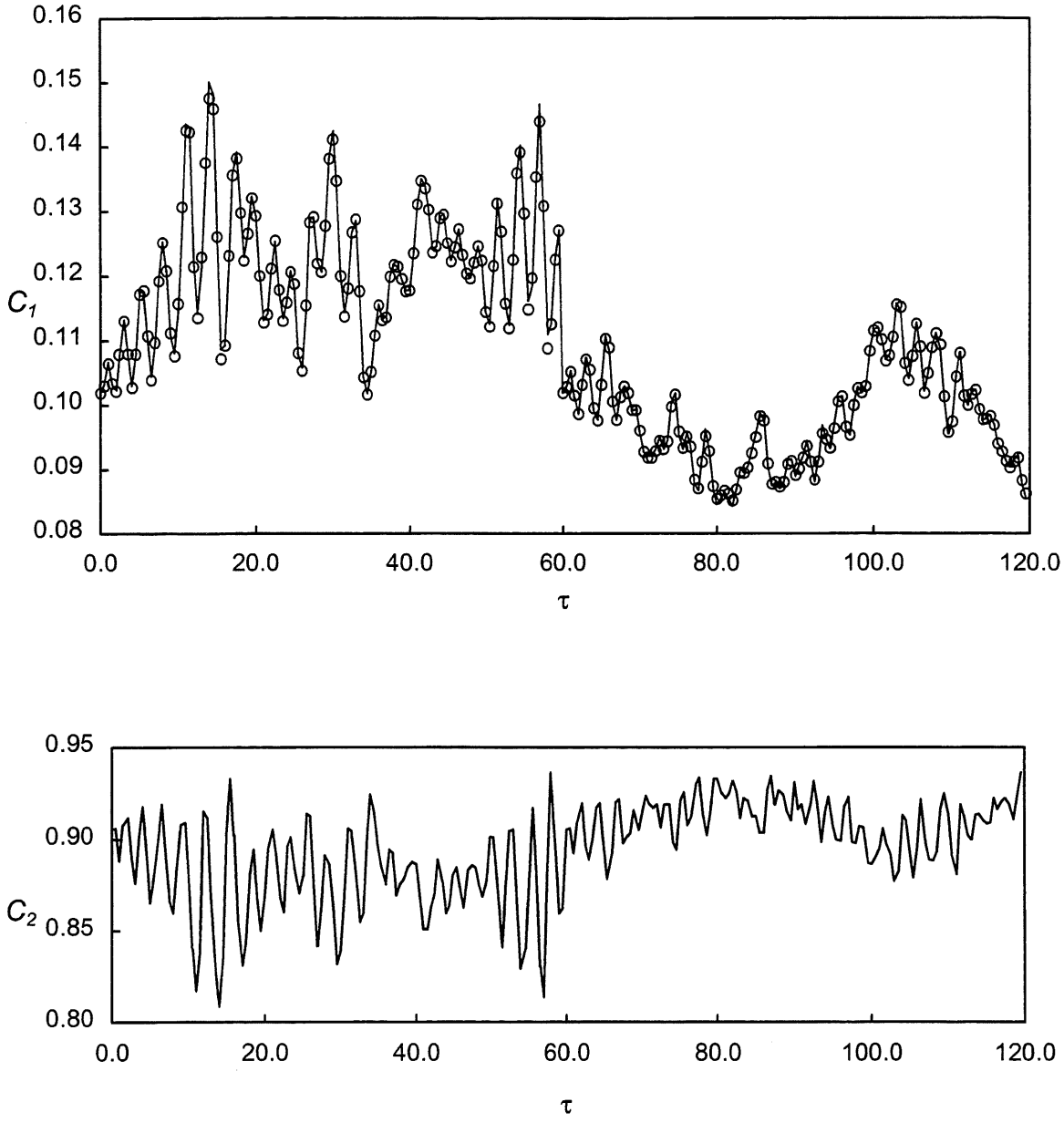


Figure 5.5 Graphiques des valeurs d'entraînement obtenues de la réponse du système à l'excitation du signal *Da* de la figure 5.4. Pour C_1, la valeur obtenue à la fin de la minimisation est aussi tracée.

———— : C_1 valeur désirée

ooooooo : C_1 valeur obtenue

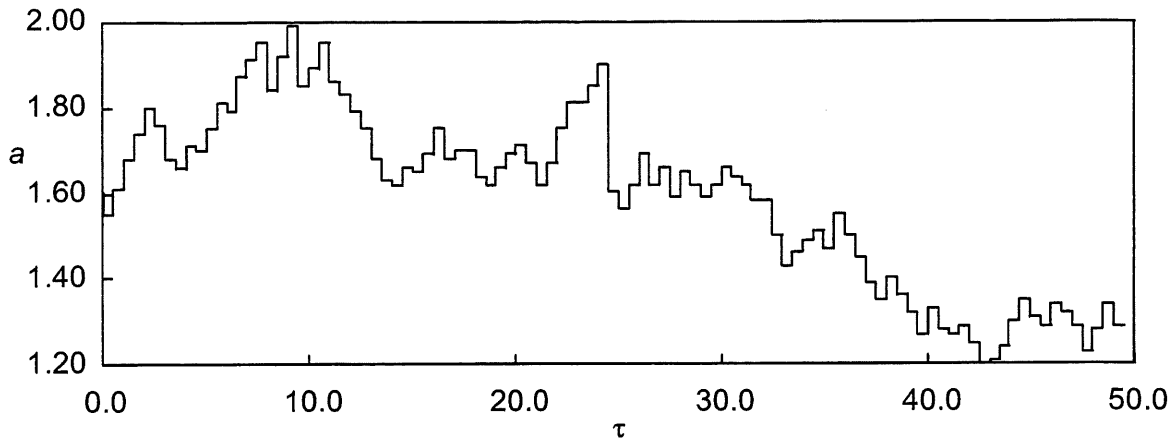


Figure 5.6 Signal de la variable manipulée *Da* utilisée pour générer les valeurs de validation de l'entraînement.

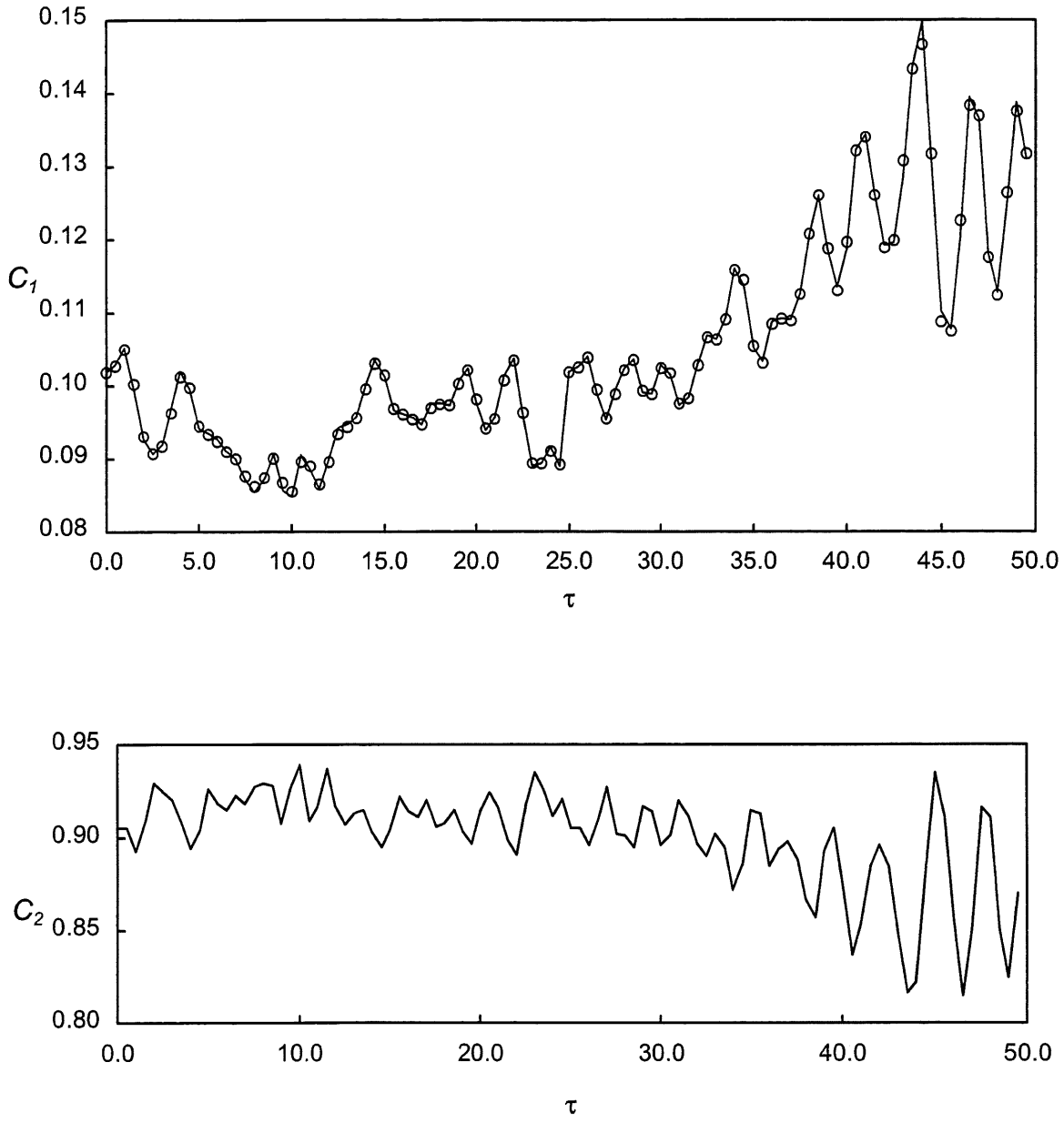


Figure 5.7 Graphique des valeurs de vérification de l'entraînement obtenues de la réponse du système à l'excitation du signal *Da* de la figure 5.6. Pour C_1, la valeur obtenue à la fin de la minimisation est aussi tracée.

———— : C_1 valeur désirée

ooooooo : C_1 valeur obtenue

5.2.4 Dynamique en boucle ouverte- réponse à l'échelon

La section 5.2.2 analyse des états d'équilibre du bioréacteur et a montré que le système étudié est complexe puisqu'il comporte plusieurs états d'équilibre possible pour une même valeur de la variable *Da* et un point de bifurcation de Hopf. Cette section s'attarde à présenter l'aspect dynamique du système lorsqu'il subit des variations de la variable manipulée *Da*.

La figure 5.8 montre la réponse du bioréacteur à un échelon qui fait passer la variable manipulée de 1.63 à 1.34. Ces valeurs correspondent à un changement des valeurs d'équilibre pour la variable C_1 de 0.1 à 0.12 et pour C_2 de 0.908 à 0.881. En étudiant la réponse du système à l'échelon, on observe qu'elle ne ressemble en rien à celle d'un système linéaire du premier ordre! Tout d'abord, c'est une réponse qui est inversée au tout début pour la variable C_1 et qui, dans cet exemple, a une durée d'inversion de $\tau = 0.35$. De plus, le comportement des variables C_1 et C_2 est oscillant pour une durée approximative de $\tau = 40$.

La figure 5.9 présente la réponse à un échelon qui fait varier *Da* de 1.34 à 1.22, c'est-à-dire très près du point de bifurcation qui se situe à $Da = 1.21$. Comme le système s'approche du point d'oscillations périodiques, la réponse du système à cet échelon est très oscillante et les oscillations persistent pour un temps supérieur à $\tau = 200$. Les nouvelles valeurs d'équilibre pour C_1 tend vers 0.131 et pour C_2 vers 0.865. Pour cet échelon, la durée de l'inversion pour la variable C_1 est de $\tau = 0.4$.

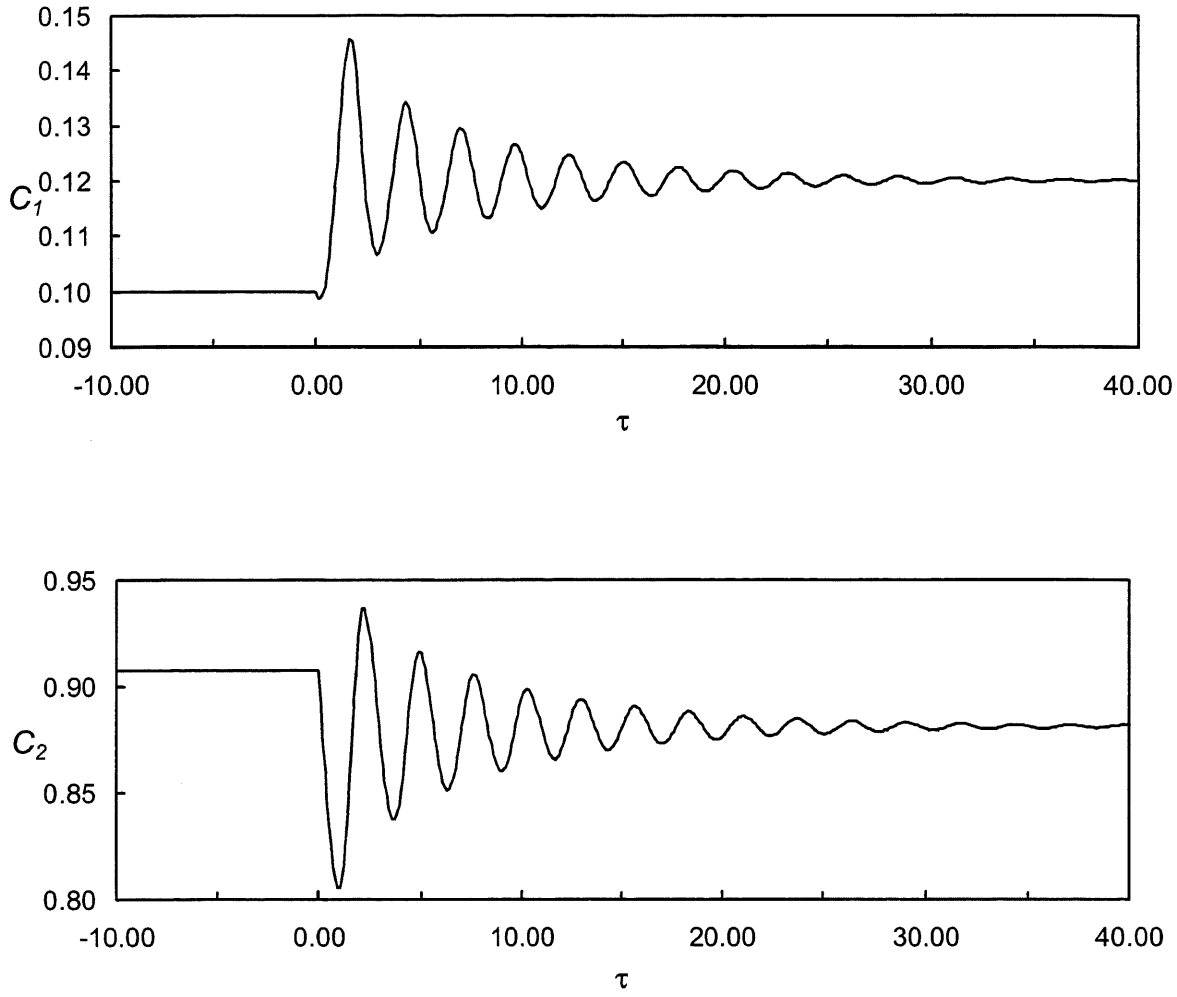


Figure 5.8 Échelon en boucle ouverte de Da = 1.63 à 1.34. Les valeurs d'équilibre correspondantes pour C_1 sont 0.1 et 0.12 et pour C_2 0.908 et 0.881.

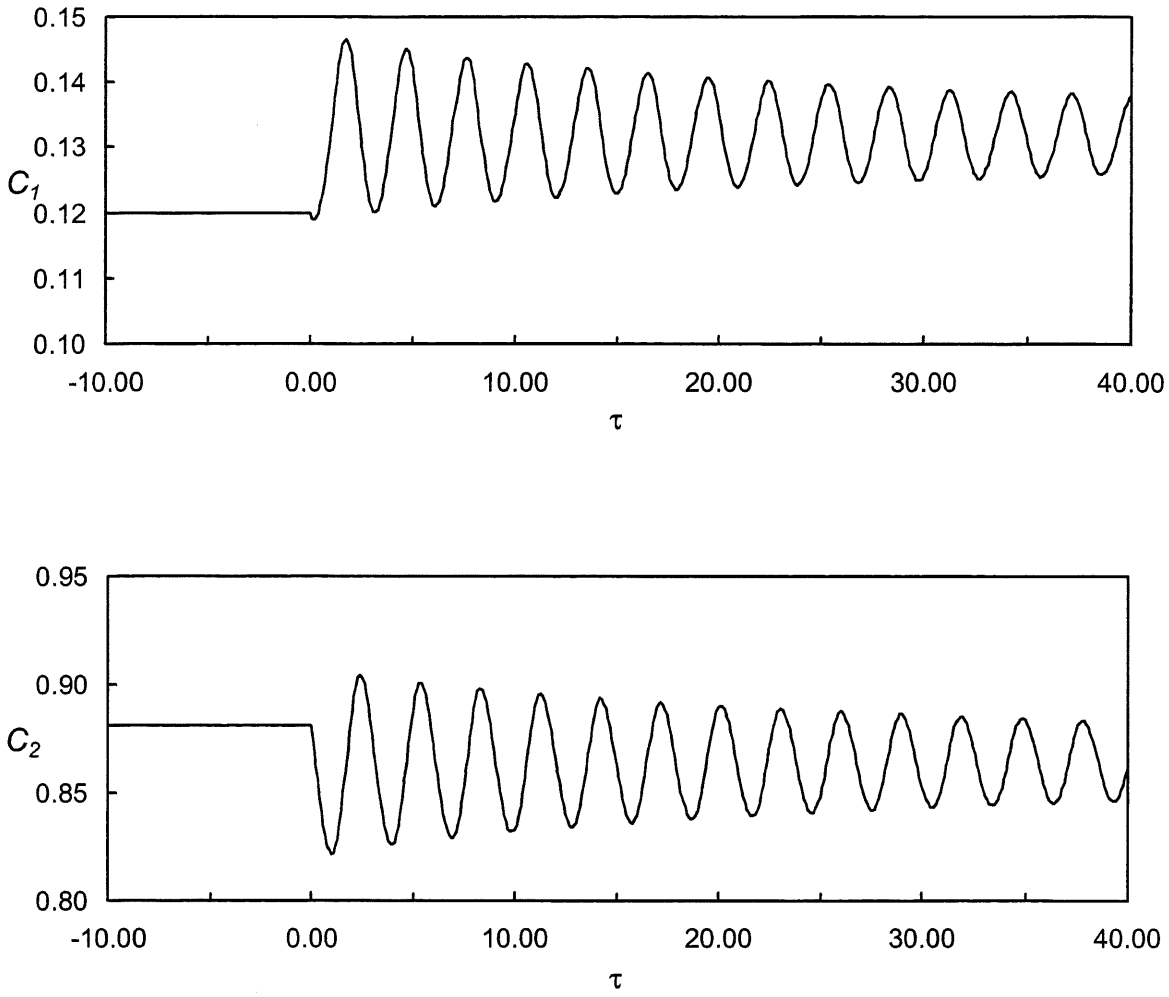


Figure 5.9 Échelon en boucle ouverte de Da = 1.34 à 1.22. Les valeurs d'équilibre correspondantes pour C_1 sont 0.12 et 0.131 et pour C_2 0.881 et 0.865.

5.2.5 Résultats du contrôle du bioréacteur

Les figures 5.10, 5.11 et 5.12 présentent le comportement du bioréacteur contrôlé par le contrôleur neuronal adaptatif lors de changements du point de consigne. Chaque simulation est effectuée en fixant le paramètre d'ajustement du contrôleur Θ à 3% , 5% et 7% respectivement.

Les différents points de consigne couvrent le domaine d'opération choisi pour le bioréacteur. Pour le temps τ inclus entre 60 et 90, la consigne est située à $C_1 = 0.13$. Cette valeur est atteinte lorsque la commande Da est à 1.22, une valeur qui est très près du point de bifurcation ($Da = 1.21$). L'analyse de la dynamique dans cette région (figure 5.9) a montré que le système a un comportement très oscillant, et l'on peut donc prévoir qu'un contrôleur, qui a des comportements brusques dans cette région, produira des oscillations dans le système.

La comparaison des trois figures (5.10, 5.11 et 5.12) conduit à la conclusion que le meilleur compromis entre la vitesse de la réponse à un changement de consigne et le comportement oscillatoire du système est obtenu lorsque Θ est égal à 0.05 (figure 5.11). Avec cette valeur, les consignes sont atteintes rapidement sans aucun dépassement significatif de la valeur. Lorsque Θ est égal à 0.03, le comportement général est similaire, mais la réponse est environ 50% plus lente. Pour une valeur de Θ de 0.07, la réponse au changement de consigne est rapide mais lorsque la consigne est près du point de bifurcation, le système oscille et des dépassements sont observés.

L'analyse de la figure 5.12 ($\Theta = 0.07$) pour τ compris dans l'intervalle [60-90] montre que le contrôleur met du temps à trouver la valeur de Da appropriée. En approchant rapidement de la consigne, le système entre en oscillation et il devient par la suite difficile d'éliminer les écarts à la consigne. On peut donc constater que dans les régions où le système a une dynamique très sensible, les différences entre le modèle et le procédé rendent l'ajustement de la variable manipulée plus difficile. Dans les situations où la dynamique est complexe et où l'on a moins confiance dans la qualité du modèle, on a tout intérêt à limiter la vitesse de variation de la variable manipulée en fixant une valeur de Θ plus faible.

Lors des changements de consigne, le système ne subit pratiquement aucune inversion apparente si l'on compare ces réponses à la réponse à un échelon de la variable manipulée. Cela s'explique par les variations sur la commande *Da* qui sont initialement faibles.

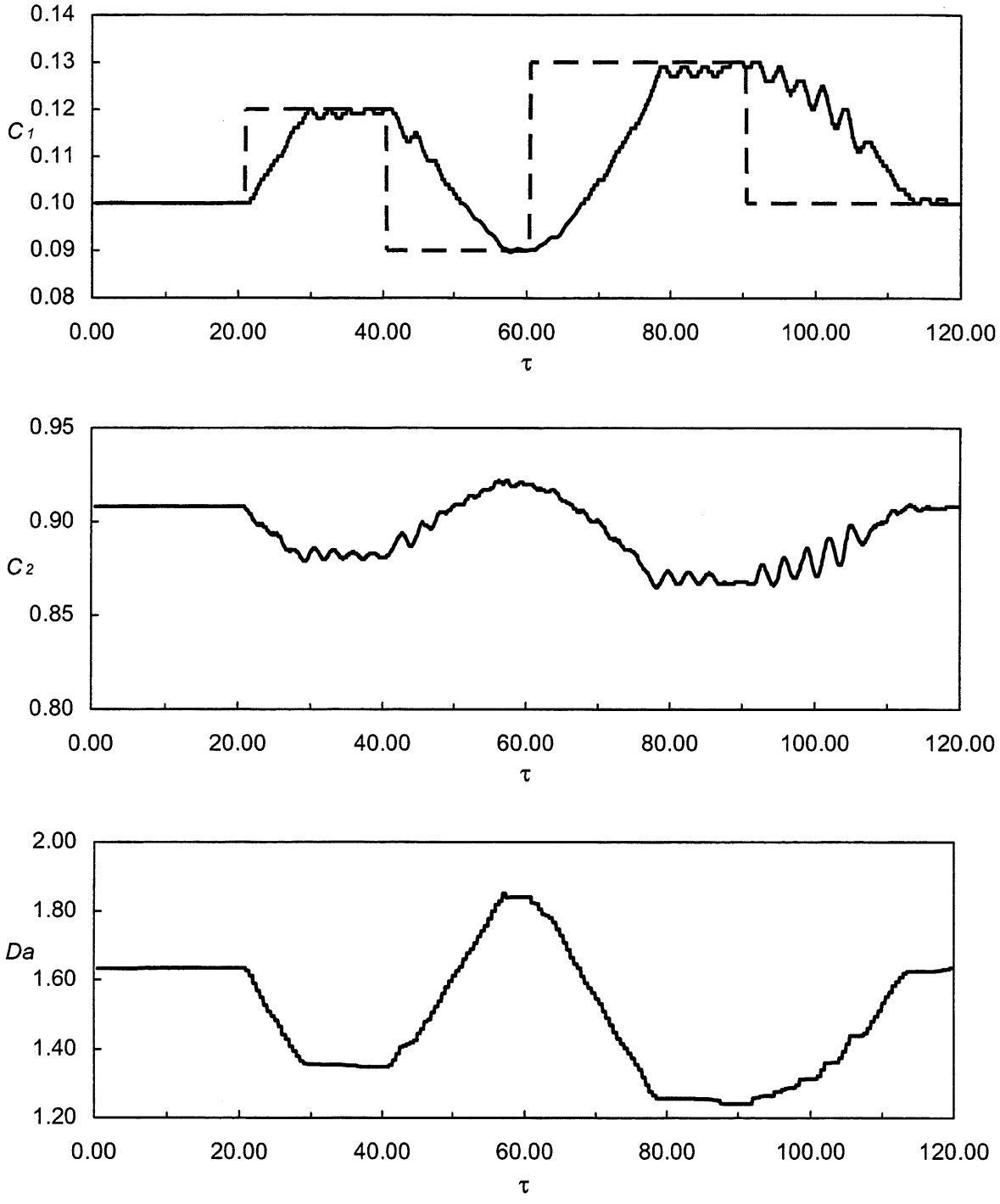


Figure 5.10 Réponse du système pour des changements de consigne avec $\Theta = 0.03$.

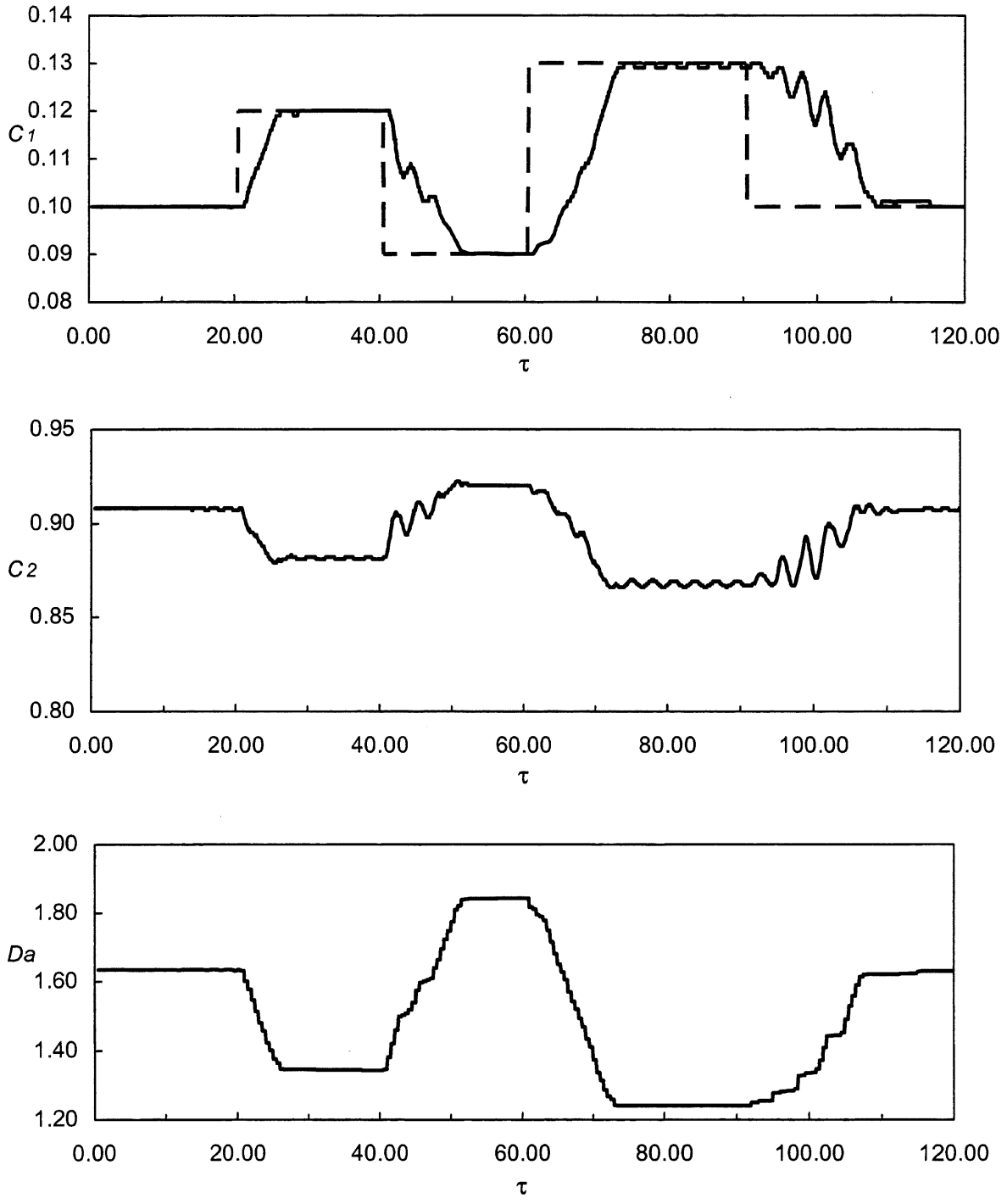


Figure 5.11 Réponse du système pour des changements de consigne avec $\Theta = 0.05$.

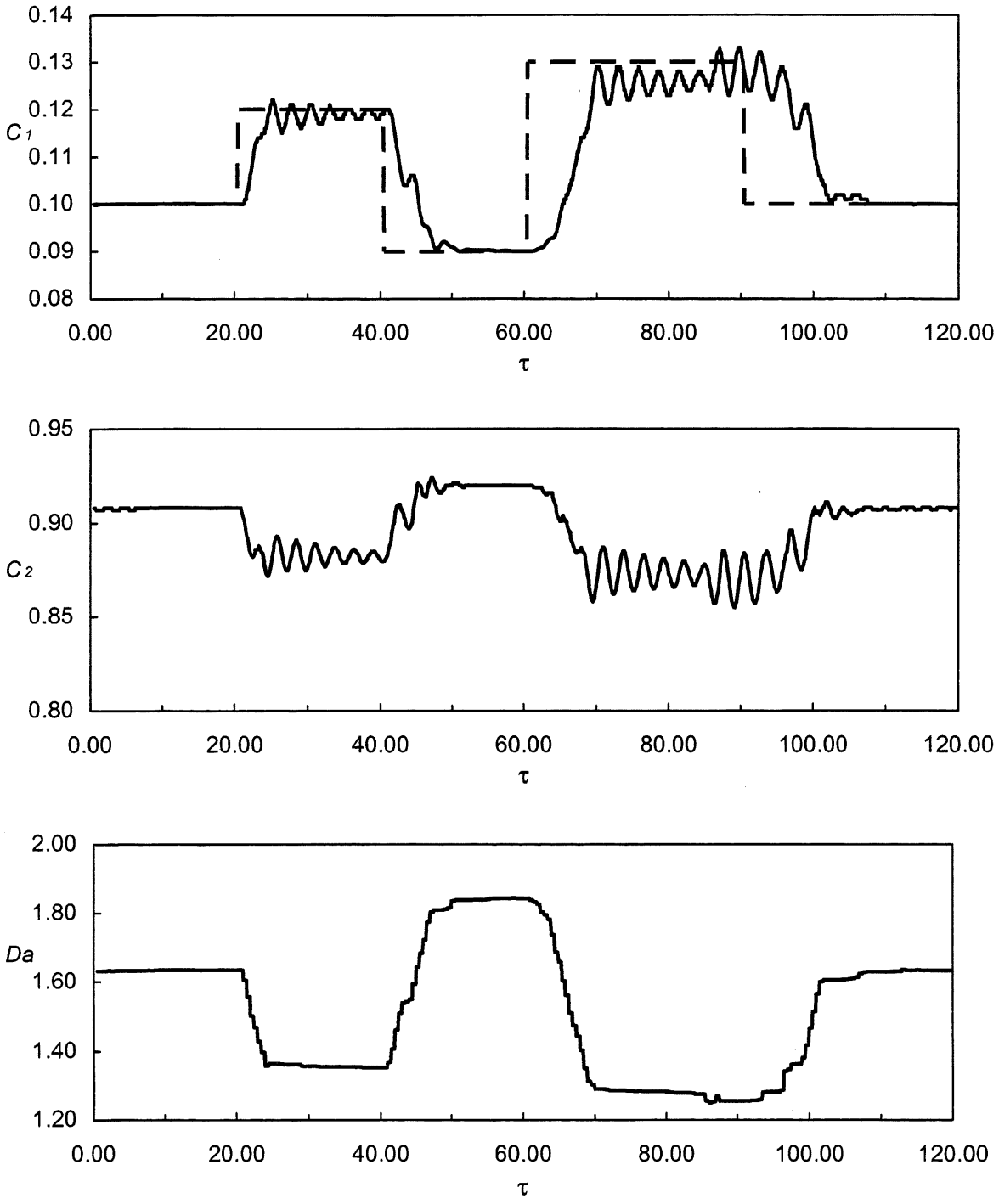


Figure 5.12 Réponse du système pour des changements de consigne avec $\Theta = 0.07$.

Les trois prochaines figures (5.13, 5.14 et 5.15) montrent le comportement du système lors de perturbations. Les perturbations sont introduites par des variations du paramètre γ. Une fois la perturbation introduite, il existe un écart entre le modèle par réseau de neurones et le procédé. Les performances du contrôleur sont aussi étudiées lors de changements de consigne en présence de tels écarts.

À la figure 5.13, le paramètre γ est varié de -2%, passant de 0.48 à 0.47. Le paramètre Θ est égal à 0.05. La perturbation a lieu à $\tau = 10$ et l'on voit qu'elle est rapidement absorbée, la commande du contrôleur passant de 1.63 à 1.58 rapidement. Le contrôleur neuronal adaptatif s'est donc bien adapté à la nouvelle situation. Lors des changements de consigne qui suivent, le contrôleur se comporte encore très bien. Les consignes à $C_1 = 0.09$ et 0.1 sont atteintes rapidement alors que les consignes situées près du point de bifurcation, $C_1 = 0.12$ et 0.13, sont approchées avec un comportement oscillant. On constate que, dans les zones où la dynamique est plus complexe, les différences entre le modèle et le procédé ont un impact plus important sur la qualité du contrôle.

Les figures 5.14 et 5.15 montrent les réponses lorsqu'une perturbation est introduite en variant le paramètre γ de +4% de 0.48 à 0.50 à $\tau = 10$. Pour chaque simulation, Θ est fixé à 0.05 et 0.03 respectivement. Dans les deux cas, le contrôleur s'adapte bien à la perturbation et la réponse est pratiquement identique. Pour les changements de consigne qui suivent, les réponses sont semblables pour les consignes à $C_1 = 0.09$ et 0.1 avec un temps plus rapide lorsque Θ est égal à 0.05. Lorsque la consigne est à $C_1 = 0.12$ et 0.13, les meilleurs résultats sont obtenus avec la valeur Θ la plus faible, le paramètre $\Theta = 0.05$ produisant une réponse beaucoup plus oscillante.

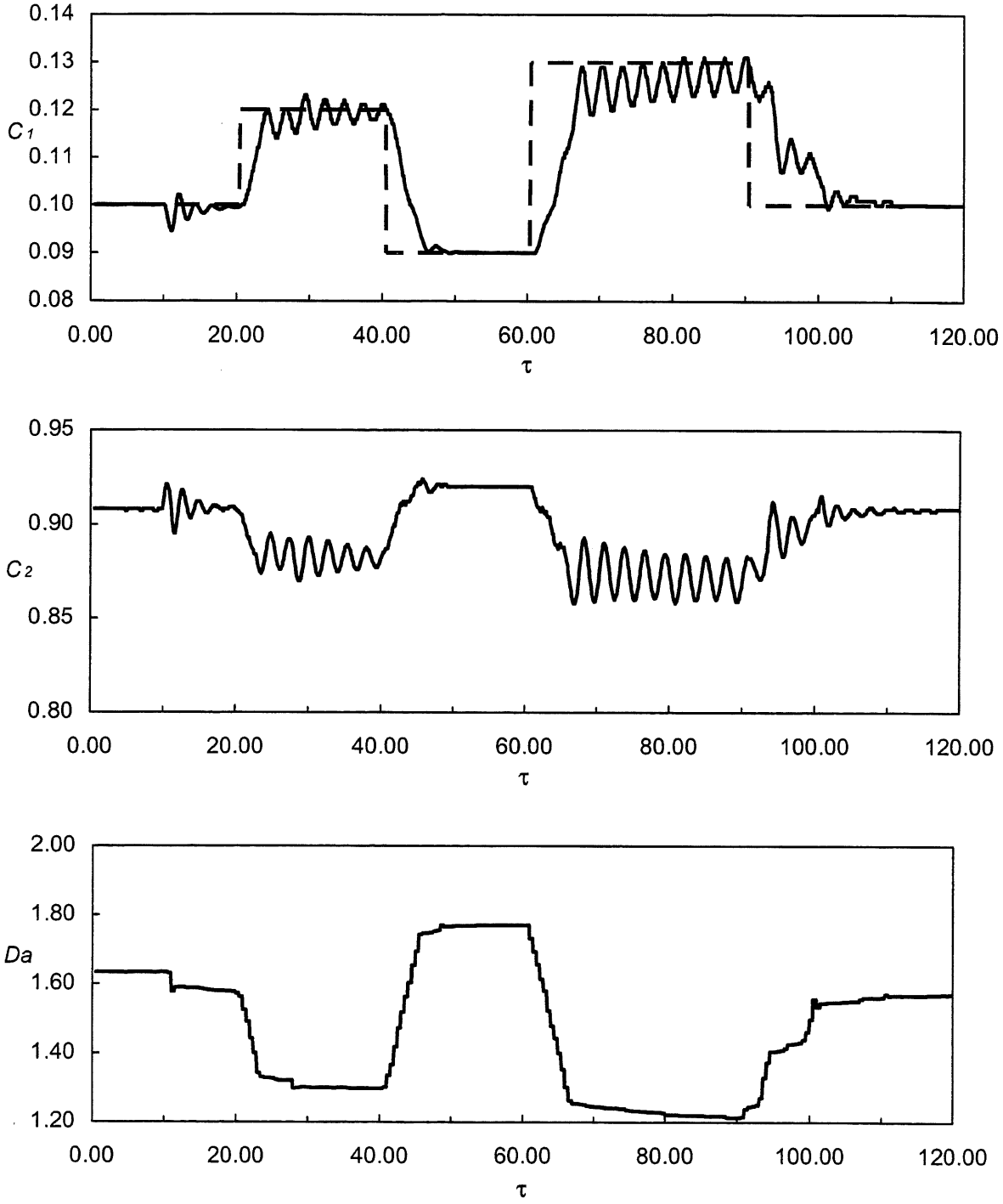


Figure 5.13 Réponse du système lors d'une perturbation. A $\tau = 10$, $\gamma = 0.48 \rightarrow 0.47$, suivie de changements de consigne ($\Theta = 0.05$).

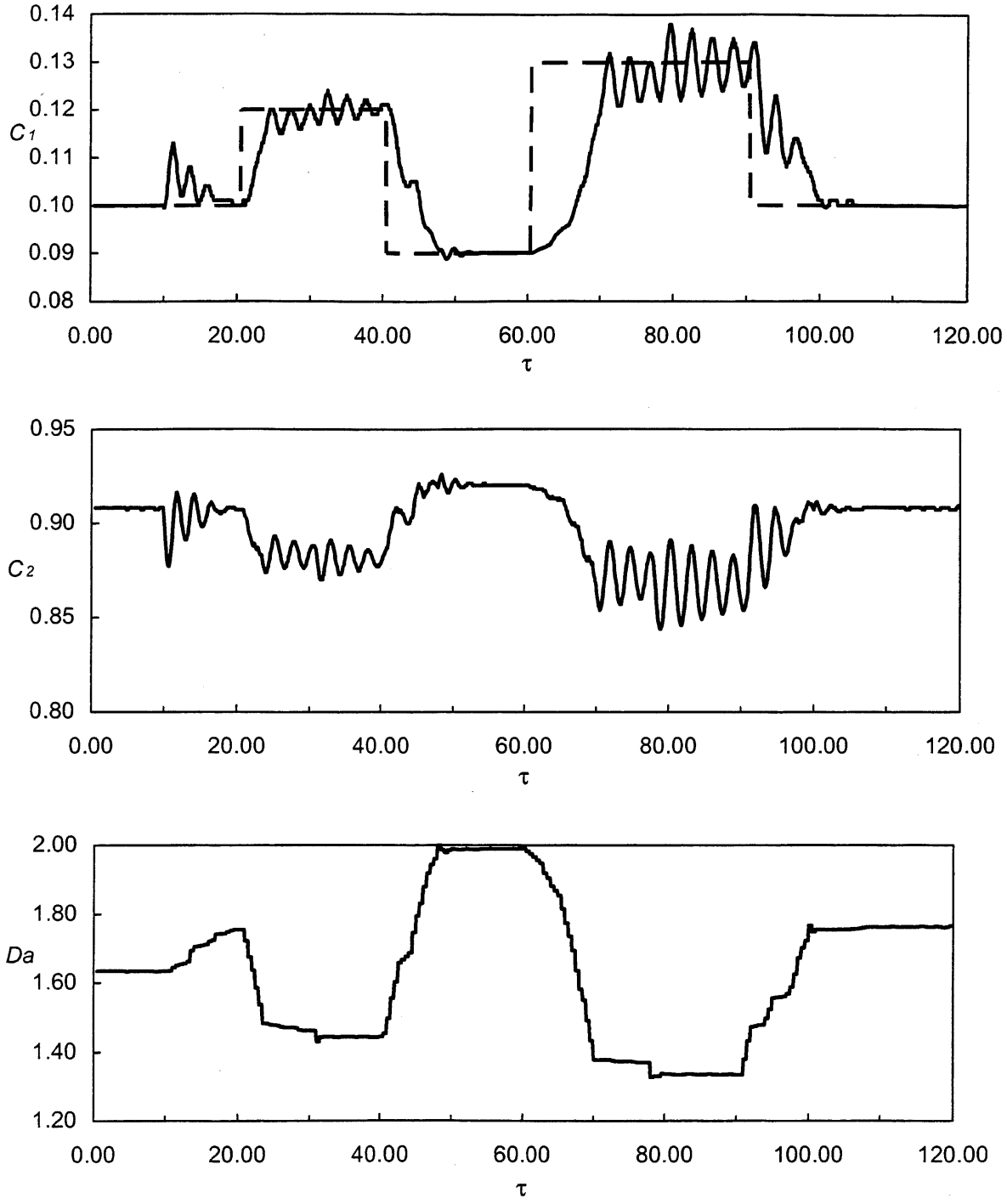


Figure 5.14 Réponse du système lors d'une perturbation. A $\tau = 10$, $\gamma = 0.48 \rightarrow 0.50$, suivie de changements de consigne ($\Theta = 0.05$).

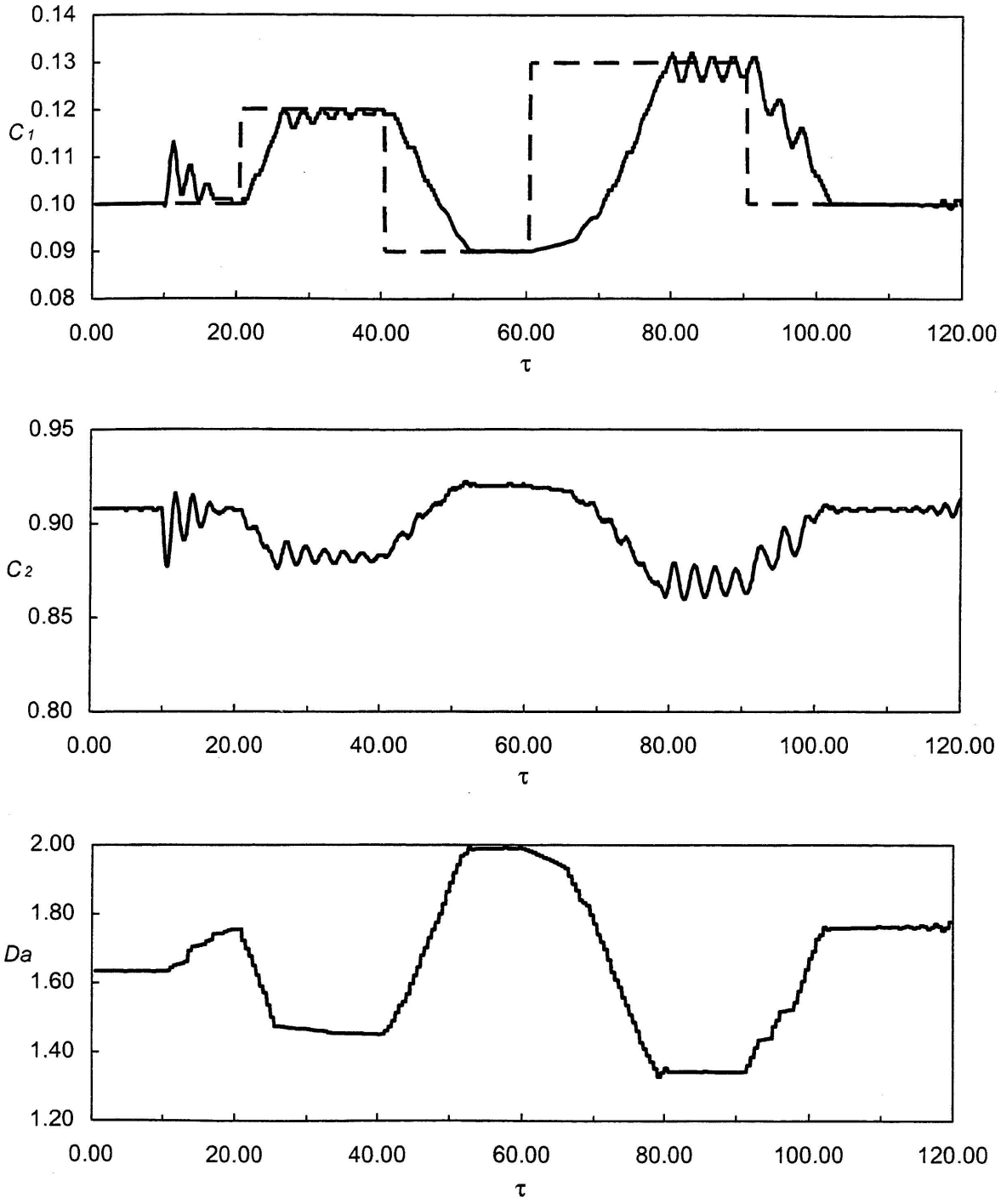


Figure 5.15 Réponse du système lors d'une perturbation. A $\tau = 10$, $\gamma = 0.48 \rightarrow 0.50$, suivie de changements de consigne ($\Theta = 0.03$).

5.3 Contrôle d'un réacteur CSTR

Le deuxième exemple étudié est un réacteur non isotherme dans lequel a lieu une réaction réversible du premier ordre. Ce système a d'abord été présenté par Economou et coll. [1986] lorsque ces derniers ont introduit le contrôle par modèle interne non linéaire (NIMC). Ils ont développé cet exemple, dont la dynamique n'est pas particulièrement complexe, afin d'illustrer les avantages du contrôle non linéaire sur le contrôle linéaire. Ils affirment qu'aucun contrôleur linéaire ne peut rivaliser en terme de performance et de robustesse avec un contrôleur non linéaire bien conçu même lorsqu'ils appliquent le NIMC à des systèmes dont les non linéarités ne sont pas très prononcées.

Ce système partage les caractéristiques d'un bon nombre d'applications en génie chimique où, à cause d'un changement de signe dans le gain, un système n'est pas « contrôlable intégralement » sur toute la plage d'opération (voir Morari et Zafiriou dans Morari et coll. [1989]). Economou et coll. [1986] ont abandonné l'exigence de l'élimination complète des écarts à la consigne pour que le système puisse être stable avec un contrôle linéaire. Ils font donc des comparaisons entre un contrôleur linéaire proportionnel et la technique NIMC qui élimine les écarts à la consigne.

Psichogios et coll. [1991] ont repris cet exemple lorsqu'ils ont comparé quatre différentes méthodes de contrôle qui utilisent des réseaux de neurones. En prenant cet exemple dans cette étude, nous pourrons comparer les résultats à ceux provenant d'autres approches qui emploient des réseaux de neurones.

5.3.1 Modèle dynamique du réacteur CSTR

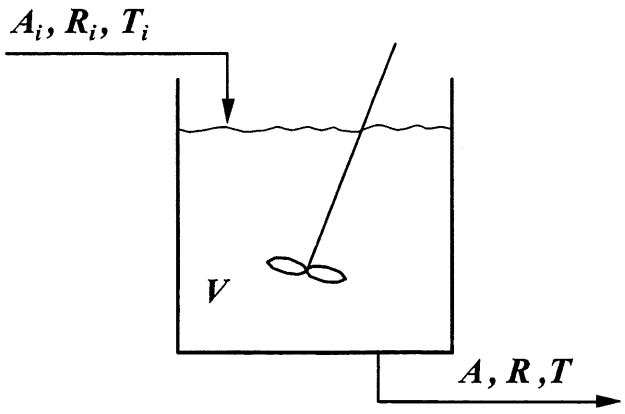


Figure 5.16 Schéma du réacteur CSTR

Le système étudié est une réaction exothermique réversible qui se produit dans un réacteur non isotherme bien mélangé en continu (CSTR) schématisé à la Figure 5.16. La réaction est du 1er ordre entre les composants A et R.

$$A \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} R \tag{5-17}$$

Le système est modélisé par un ensemble d'équations différentielles couplées. Les bilans de masse sur le réactif et sur le produit s'écrivent:

$$\frac{dA}{dt} = \frac{1}{\tau}(A_i - A) - k_1 A + k_{-1} R \tag{5-18}$$

$$\frac{dR}{dt} = \frac{1}{\tau}(R_i - R) + k_1 A - k_{-1} R \tag{5-19}$$

et le bilan d'énergie comme:

$$\frac{dT}{dt} = \left(\frac{-\Delta H_R}{\rho C_p}\right)\left(k_1 A - k_{-1} R\right) + \frac{1}{\tau}\left(T_i - T\right) \tag{5-20}$$

Les taux spécifiques de réactions sont définis par:

$$k_1 = C_1 \exp\left(\frac{-Q_1}{\Re T}\right) \tag{5-21}$$

$$k_{-1} = C_{-1} \exp\left(\frac{-Q_{-1}}{\Re T}\right) \tag{5-22}$$

Les constantes sont celles utilisées par Economou et coll. [1986] et sont présentées dans le Tableau 5.1.

TABLEAU 5.1 Design du réacteur : constantes et conditions d'opération

τ	=	60 s	$-\Delta H_R$	=	5 000 cal mol^{-1}
C_1	=	5 x 10^3 s^{-1}	ρ	=	1 kg/L
C_2	=	1 x 10^6 s^{-1}	C_p	=	1 000 cla kg^{-1} K^{-1}
Q_1	=	10 000 cal mol^{-1}	A_i	=	1.0 mol/L
Q_{-1}	=	15 000 cal mol^{-1}	R_i	=	0.0 mol/L
$\Re$	=	1.987 cal mol^{-1} K^{-1}			

Les concentrations A_i et R_i à l'alimentation du réacteur sont considérées constantes. La température T_i du débit d'alimentation est la variable manipulée u et la concentration R du produit à la sortie et la variable contrôlée y pour le système.

5.3.2 Analyse de l'état d'équilibre du système

Cette section présente une analyse des états d'équilibre du système. Le système est étudié pour les valeurs de A_i = 1.0 mol/L et R_i = 0.0 mol/L. La figure 5.17 présente la courbe de

la concentration R du produit en fonction de la température T_i du débit d'entrée dans le réacteur pour un système en état de régime.

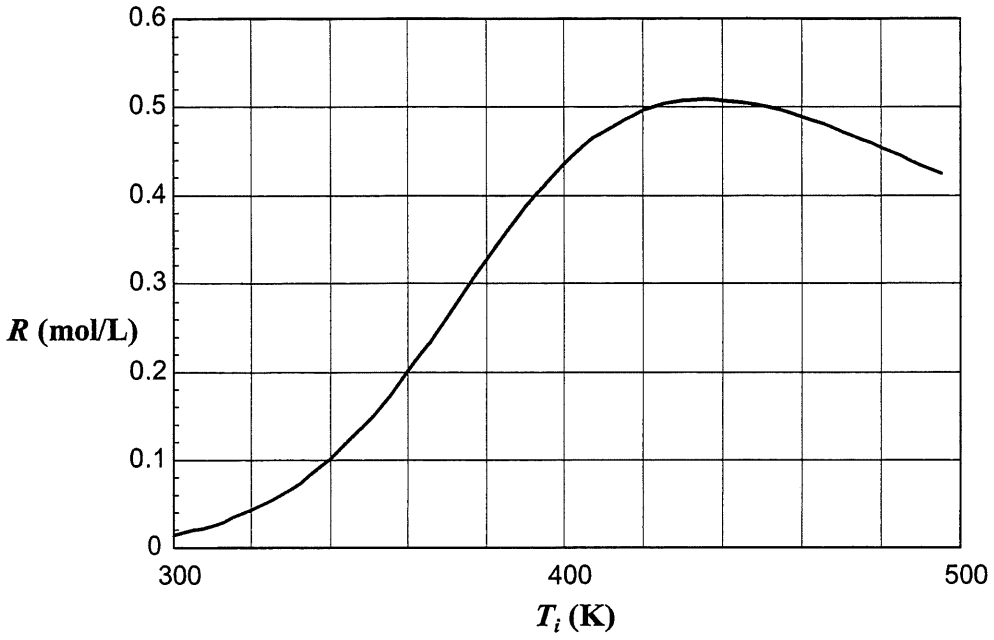


Figure 5.17 Courbe de la concentration R du produit en fonction de la température T_i du débit d'entrée dans le réacteur pour un système en état de régime.

En boucle ouverte, ce système est toujours stable quelles que soient les conditions initiales. L'analyse de la figure 5.17 montre que la relation entre la variable contrôlée et la variable manipulée de ce système est non linéaire et que la conversion en produit R atteint un maximum de 0.5085 lorsque T_i est égal à 434 K. A ce point, le système n'est pas inversible (le gain est zéro). De plus, ce système n'est pas biunivoque car la courbe entrée-sortie à l'état de régime n'est pas inversible (il y a deux températures qui peuvent produire une même conversion).

5.3.3 Modélisation de la dynamique du réacteur par réseaux de neurones

Dans un système SISO, il est commun que seules la variable contrôlée et la variable manipulée soient mesurées. Dans cet exemple, afin de reproduire des conditions réalistes et en même temps plus difficiles pour le contrôleur, la modélisation de la dynamique du procédé par réseau de neurones est faite uniquement à partir de valeurs de R, la variable contrôlée, et de T_i, la variable manipulée. Lorsqu'ils ont modélisé ce système Psichogios et coll. [1991] ont utilisé deux approches, une qui suppose que toutes les variables d'états sont mesurées et une autre qui utilise uniquement des données provenant des entrées-sorties du système. Ils ont utilisé le modèle provenant de la première approche avec les quatre méthodes de contrôle qu'ils ont essayées et le modèle qui provient de la deuxième approche avec la méthode par modèle interne (IMC) où le réseau est inversé à chaque itération. La comparaison des résultats obtenus par les deux approches montre des temps de réponse équivalents mais avec un système en boucle fermée plus oscillant au niveau de la commande pour la deuxième approche. Ce résultat est conforme à la théorie du contrôle par modèle interne qui dit qu'en augmentant la qualité du modèle, on diminue le travail du contrôleur.

Le vecteur d'entrée du réseau de neurones modélisant la dynamique du procédé est établi selon les critères utilisés par la technique ARMA. Comme il y a trois variables d'états, les trois itérations précédentes de la variable manipulée et de la variable contrôlée sont utilisées à l'entrée du réseau et la sortie est la prédiction de la concentration à la prochaine itération.

$$\hat{R}(t+1) = F\left[R(t), R(t-1), R(t-2), T_i(t), T_i(t-1), T_i(t-2)\right] \qquad (5\text{-}23)$$

Le réseau contient une seule couche cachée de dix neurones. La période d'échantillonnage Δt est fixée à 30 secondes ce qui est la moitié du temps de séjour du réacteur τ égal à 60 secondes. Cette valeur est la même que celle utilisée par Psichogios et coll. [1991] (Economou et coll. [1986] n'ont pas indiqué la période qu'ils ont utilisée).

L'objectif du contrôle est d'opérer le réacteur près du point de conversion maximale tout en maintenant le système en boucle fermée stable face à des perturbations. Les résultats de Economou et coll. [1986] montrent qu'un contrôleur linéaire proportionnel est incapable de maintenir le système stable en présence de perturbations lorsque la consigne est près du maximum.

La figure 5.18 présente les états d'équilibre de la figure 5.17 en ajoutant les frontières du domaine d'opération modélisé et les valeurs du point optimal d'opération. La température T_i à l'entrée du réacteur est incluse entre 380 et 440 K. Le domaine encadre le point optimal d'opération mais se concentre principalement sur une zone où il existe une relation unique entre l'entrée et la sortie.

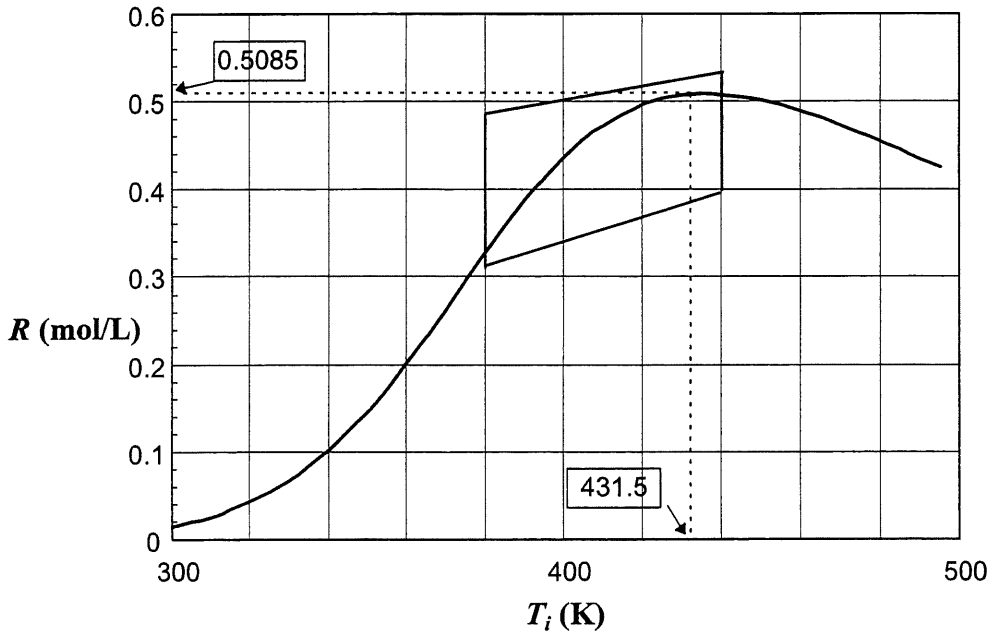
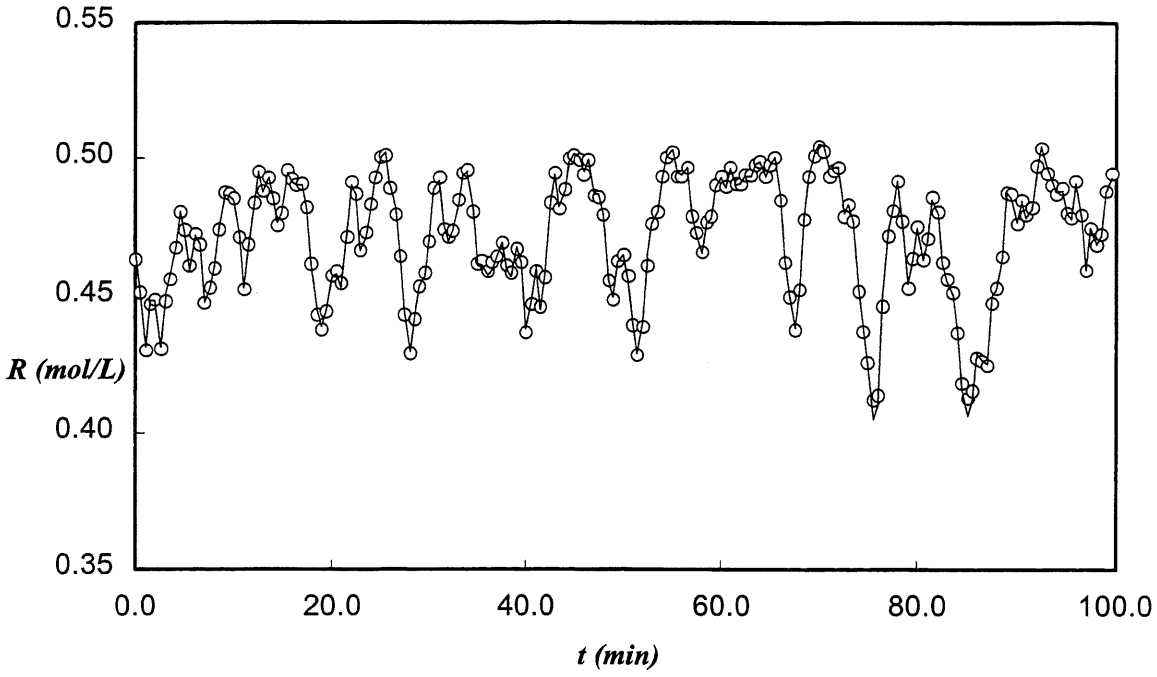


Figure 5.18 Graphique de la région modélisée du réacteur. La plage encadrée représente approximativement le domaine d'opération dynamique qui est modélisé.

Les valeurs d'entraînement pour le modèle sont générées en intégrant les équations d'états (5.18), (5.19) et (5.20) autour d'un signal T_i à partir d'un point initial de 410 K et des

conditions d'équilibre correspondantes. Le signal est varié aléatoirement avec une distribution uniforme entre 380 et 440 K pour T_i ce qui représente approximativement ± 9% de la valeur médiane. L'ensemble d'entraînement est constitué de deux cents présentations. Les valeurs de l'ensemble de validation sont obtenues de façon différente. Elles proviennent de deux séries de trente valeurs obtenues en intégrant les équations autour d'un signal de T_i = 400 K et 420 K respectivement et qui est varié de ± 2% avec une distribution uniforme. Cette procédure a pour but de s'assurer que bien que le modèle soit entraîné à partir de grandes variations de T_i sur l'ensemble du domaine, il a bien capté la dynamique des petites variations de la commande. Cette confirmation est particulièrement importante dans la deuxième série de valeurs qui se trouve très près du point de conversion optimale.

La figure 5.19 présente les valeurs d'entraînement et le résultat de l'apprentissage après 300 itérations et la figure 5.20 présente les valeurs de validation. A ce stade, la fonction erreur (équation 2.6) divisée par le nombre P de valeurs est de $1.1\text{x}10^{-4}$ pour les valeurs d'entraînement et de $1.2\text{x}10^{-5}$ pour les valeurs de validation. Si l'on poursuit l'optimisation, l'erreur calculée pour les valeurs de validation augmente. L'ordre de grandeur sur l'erreur entre les deux séries peut paraître curieux de prime abord puisqu'il favorise les valeurs de validation. Dans ce contexte, il faut se rappeler que les séries sont obtenues de deux méthodes différentes et que, dans ce cas, rien ne garantit que l'erreur sera nécessairement plus faible pour les valeurs d'entraînement.



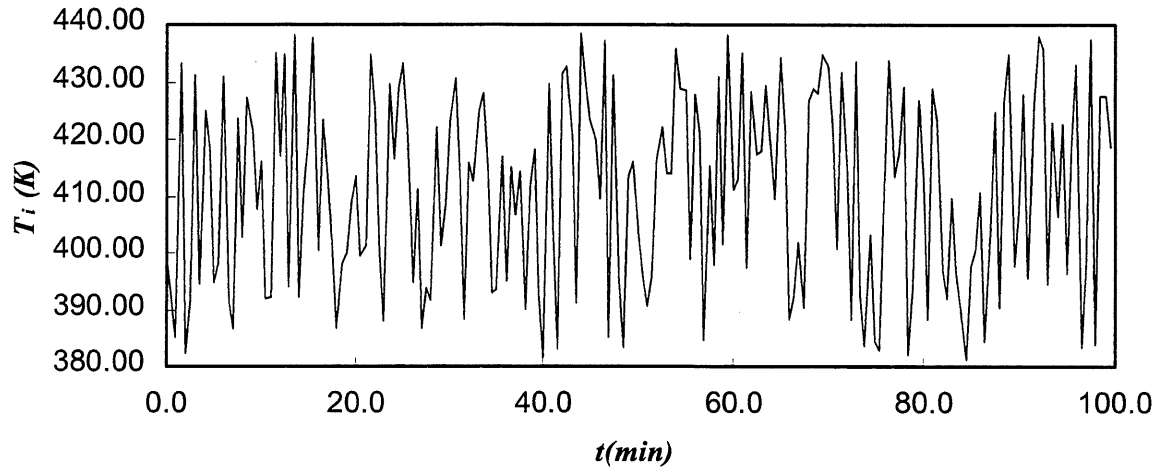
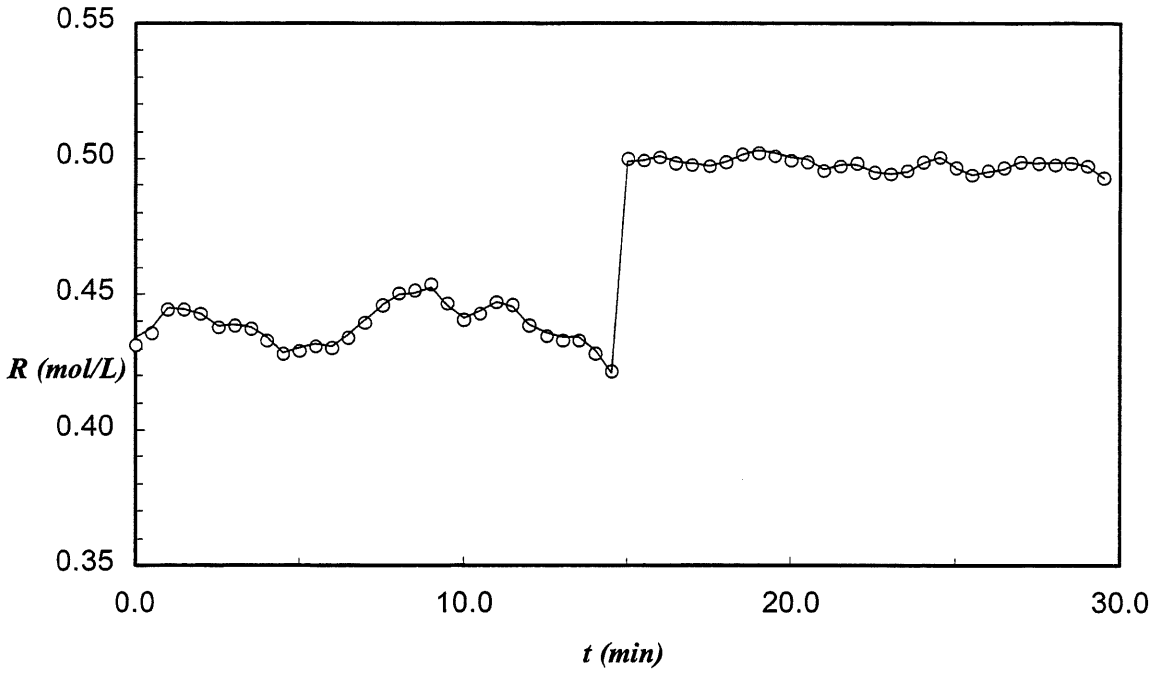


Figure 5.19 Valeurs d'entraînement pour le réacteur CSTR. Pour *R*, la valeur obtenue à la fin de la minimisation est aussi tracée.

———— : *R* valeur désirée

ooooooo : *R* valeur obtenue



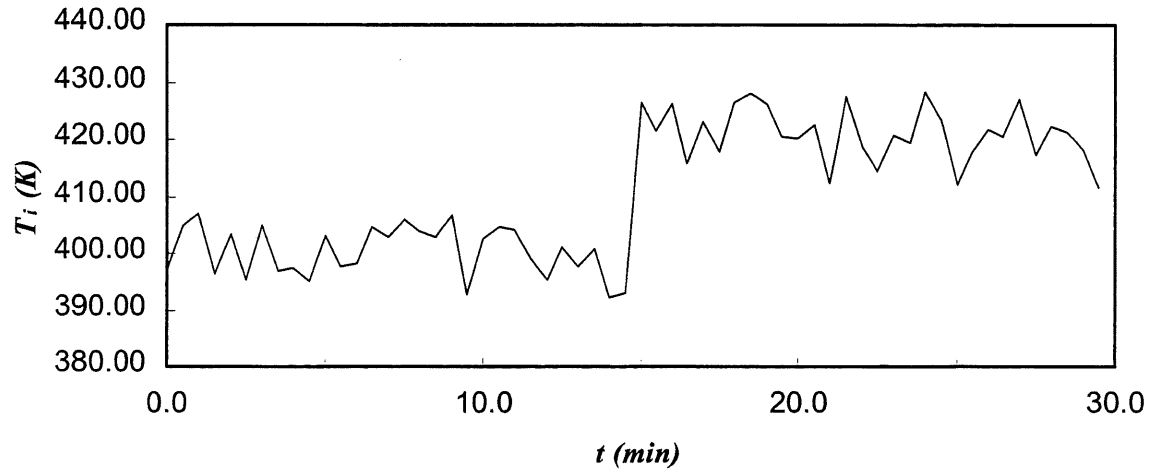


Figure 5.20 Valeurs de validation pour le réacteur CSTR. Pour R, la valeur obtenue à la fin de la minimisation est aussi tracée.

———— : R valeur désirée

ooooooo : R valeur obtenue

5.3.4 Résultats du contrôle du réacteur CSTR

Les figures 5.21 et 5.22 présentent la réponse du système lors de changement de consigne lorsque le paramètre Θ du contrôleur neuronal adaptatif est fixé à 0.03 et 0.05 respectivement. Dans cet exemple, la variable manipulée T_i est bornée entre 381 et 439 K, bornes qui sont légèrement à l'intérieur du domaine d'apprentissage du modèle par réseaux de neurones.

Pour la période incluse entre 20 et 60 minutes, la consigne est établie à $R = 0.505$, une valeur qui est près du point optimal de conversion ($R = 0.508$). Le second point de consigne est à $R = 0.45$.

Les résultats sont très bons lorsque Θ est égale à 0.03, on obtient une réponse rapide, peu de dépassement de consigne et peu d'oscillation. Lorsque Θ est égal à 0.05, la réponse est évidemment plus rapide et est excellente pour le premier point de consigne à $R = 0.505$. Pour le second point de consigne à $R = 0.45$, on observe un dépassement initial assez important avec $R = 0.44$ suivi d'oscillations autour de la consigne qui sont causées par des réactions trop brusques et trop importantes du contrôleur neuronal adaptatif aux écarts par rapport la consigne. Cette valeur du paramètre Θ à 0.05 est donc trop grande.

Ces résultats démontrent que l'on peut très bien utiliser un modèle établi à partir de données partielles du procédé (valeurs précédentes des variables contrôlées et manipulées) et obtenir de bonnes réponses.

Les deux figures suivantes, 5.23 et 5.24, reprennent l'exemple utilisé par Psichogios et coll. [1991]. Durant 20 minutes, de t = 20 à 40 min., le système subit une perturbation de -2% dans la concentration A_i à l'entrée du réacteur et le contrôleur doit maintenir la consigne de $R = 0.505$. La perturbation est assez importante pour que le système, qui est déjà près du point de conversion maximale, ne puisse atteindre la consigne. La réponse optimale du contrôleur à cette perturbation est de positionner la température d'entrée au point maximal de conversion, $T_i =$

431.5 K, de façon à minimiser l'écart à la consigne et de reprendre la position initiale T_i = 427 K à la fin de la perturbation. Il faut remarquer que, si aucun filtre, comme utilisé par Economou et coll. [1986], ou autre limitation n'est imposé au contrôleur, le système peut tout aussi bien se positionner de l'autre coté du point de conversion maximale à une valeur T_i qui se situe à 458 K, et qui procure la même conversion.

La figure 5.23 donne la réponse du contrôleur neuronal adaptatif à cette perturbation lorsque Θ est égal à 0.03. Initialement, la variable R diminue et le contrôleur s'adapte en augmentant la température. Lorsque la valeur de R commence à se stabiliser, le contrôleur entraîne le système vers la borne supérieure située à 439 K. Lorsque la perturbation est terminée, le système est incapable de retrouver la consigne puisque qu'il ne peut passer par-dessus le point maximal situé à T_i = 431.5 K.

Psichogios et coll. [1991] n'ont pas rencontré cette difficulté dans leur exemple puisqu'ils ont borné le domaine du modèle par réseau de neurones au point optimal de façon à pouvoir bien limiter le contrôleur face à une perturbation sans utiliser de filtre dans la structure de contrôle par modèle interne. De plus, comme leur approche demande d'inverser à chaque itération le modèle par réseau de neurones, ils devaient borner le modèle dans une zone où l'inverse est unique.

La figure suivante 5.24 reprend la même simulation mais en fixant la borne supérieure de la variable T_i au point maximal de conversion. Cette fois le système, après avoir rencontré la borne durant la perturbation, reprend la bonne valeur après celle-ci. Cette réponse est similaire à celle obtenue par Psichogios et coll. [1991] dans l'exemple de contrôle par un modèle interne établi à partir des variables contrôlées et manipulées. Elle est aussi légèrement oscillante à la fin de la perturbation et puisque le contrôleur ne réagit pas tant que la consigne n'est pas dépassée, la conversion maximale de R = 0.508 est pratiquement atteinte.

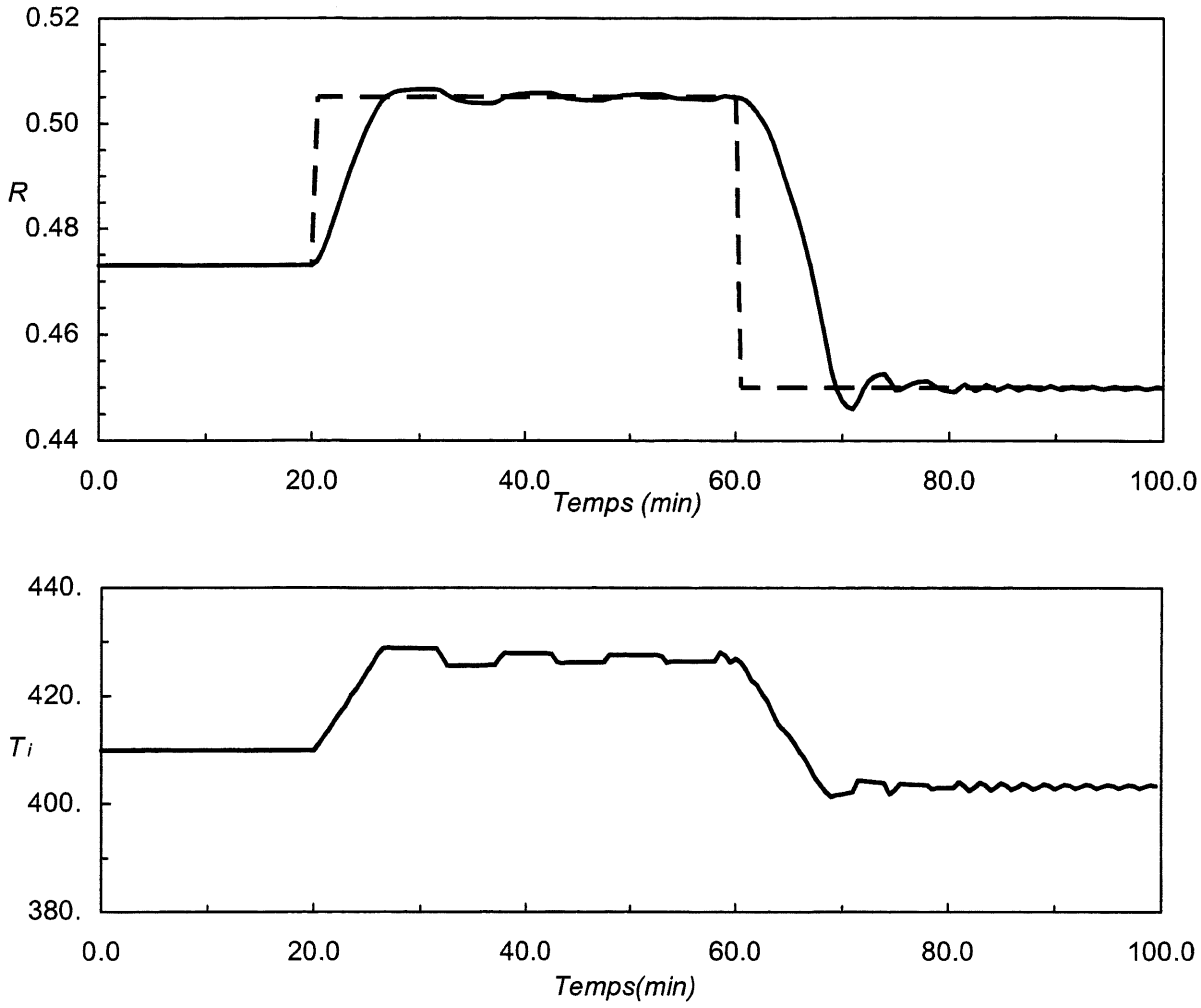


Figure 5.21 Réponse du système pour des changements de consigne avec $\Theta = 0.03$

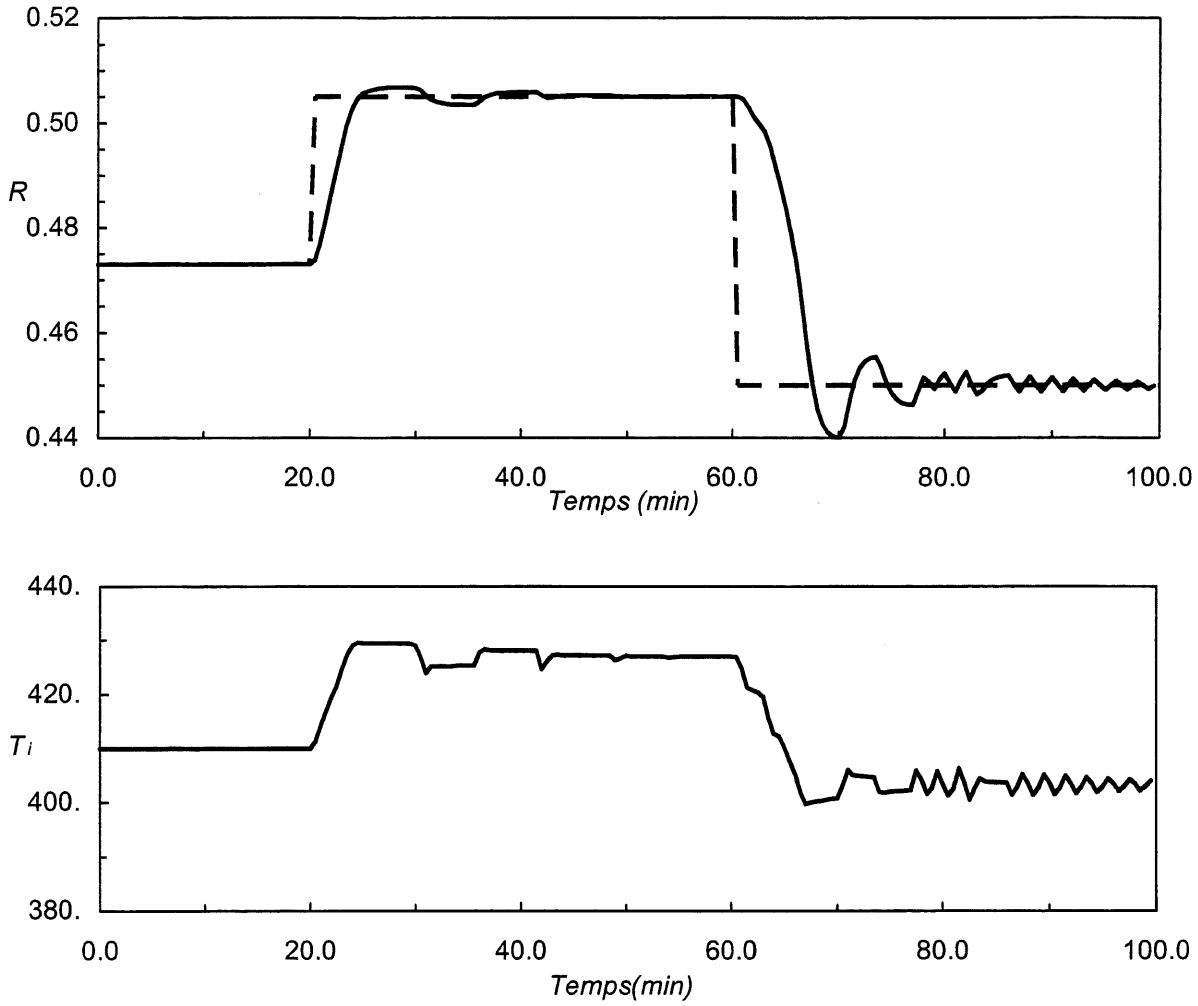


Figure 5.22 Réponse du système pour des changements de consigne avec $\Theta = 0.05$

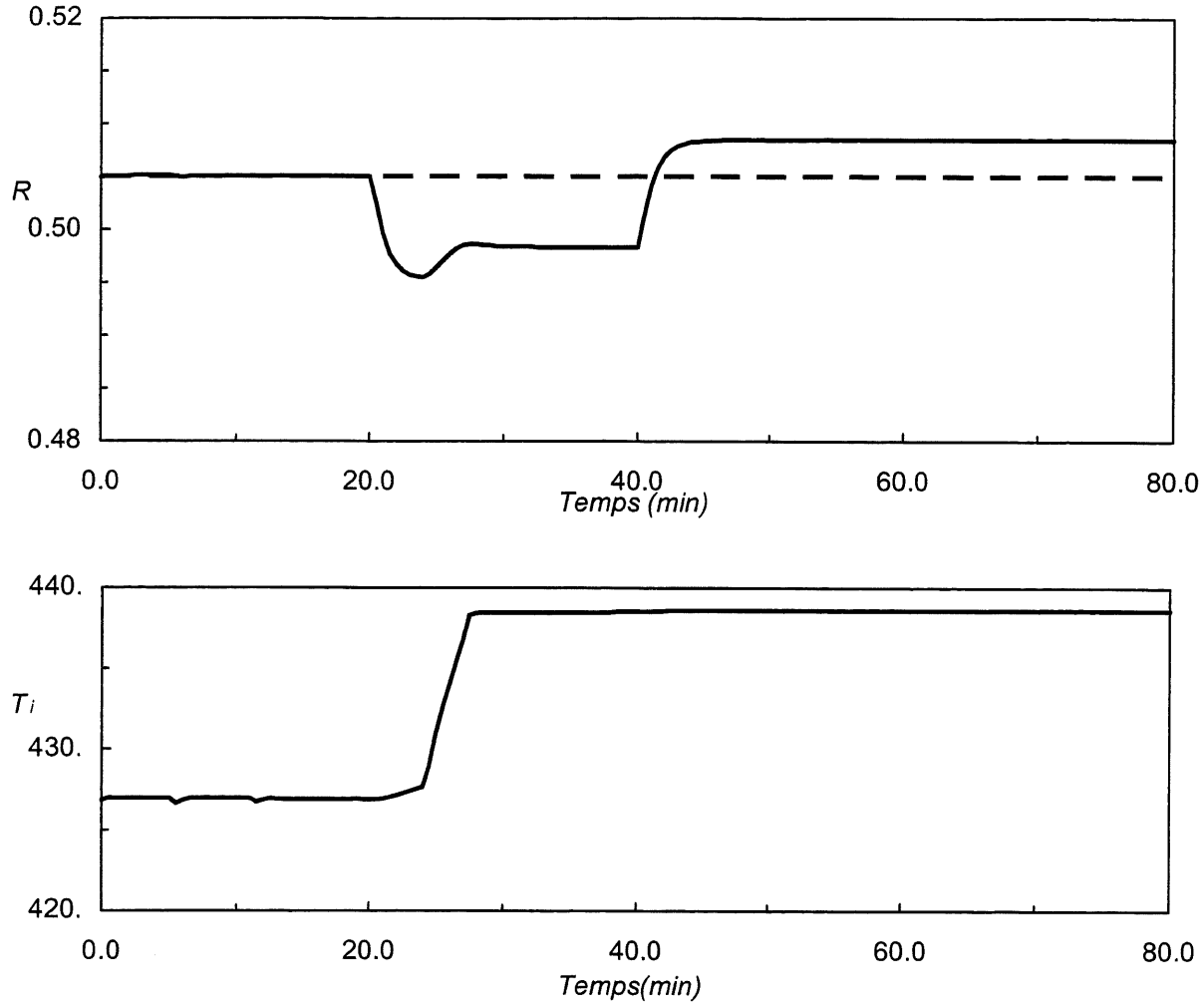


Figure 5.23 Système face à une perturbation de -2% de A_i entre t = 20 et t = 40 min.

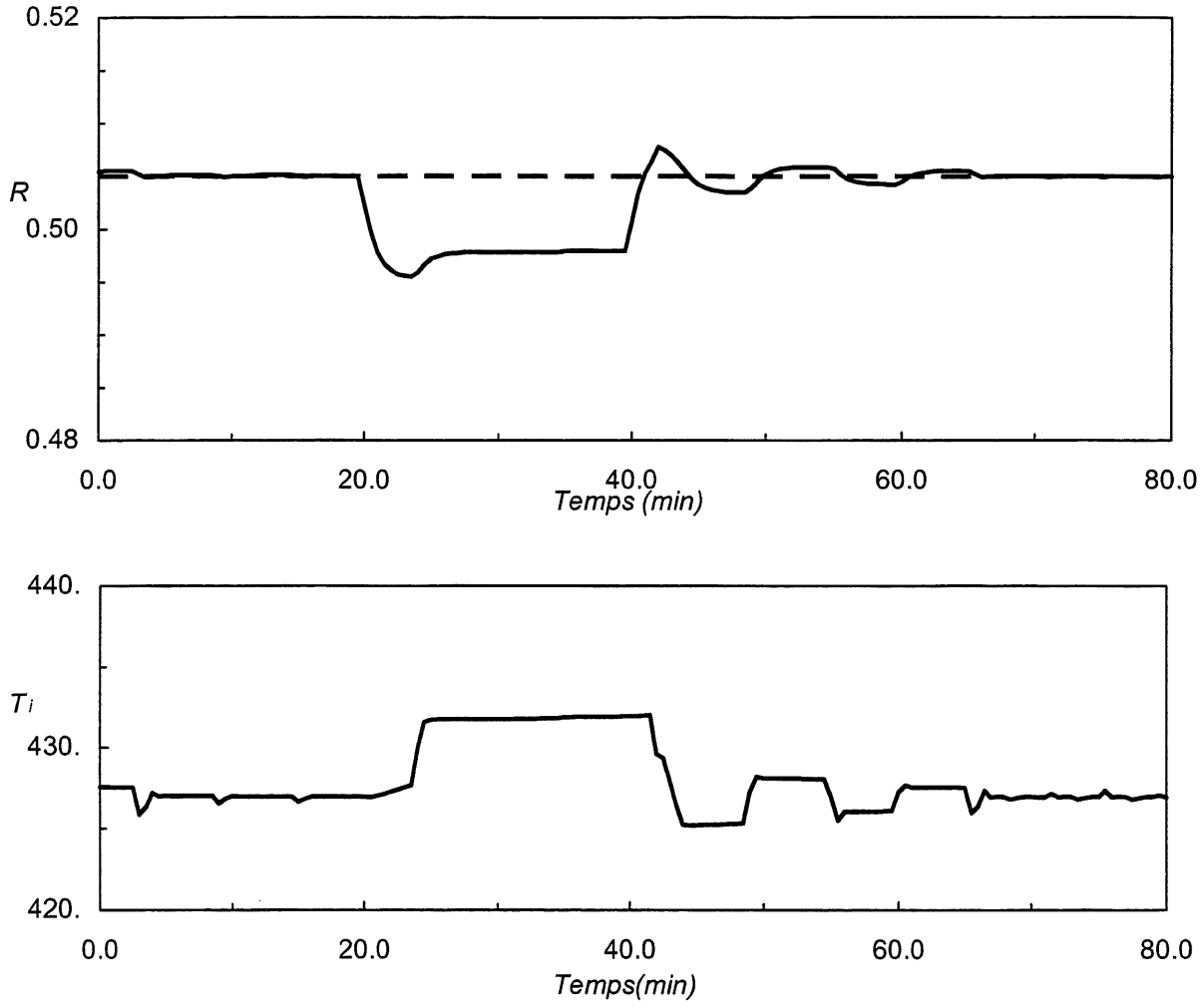


Figure 5.24 Système face à une perturbation de -2% de A_i entre t = 20 et t = 40 min lorsque $T_i$max = 431.5.

CONCLUSION

Ce travail a développé un algorithme de contrôle adaptatif constitué uniquement de réseaux de neurones. Cet algorithme utilise un entraînement spécialisé de réseaux de neurones. Par rapport aux autres approches qui utilisent un entraînement spécialisé, celle-ci se démarque par l'emploi d'un réseau de neurones distinct pour modéliser la dynamique du procédé, par une utilisation cohérente des coordonnées de temps et par une nouvelle fonction erreur à minimiser qui est le reflet de la composante prédictive de l'algorithme.

La configuration développée est constituée de deux réseaux de neurones : le premier est entraîné comme modèle dynamique du procédé ; et le second, appelé contrôleur neuronal adaptatif, est un modèle inverse de la dynamique du procédé et est adapté en continu. Il y a un avantage à utiliser deux réseaux de neurones distincts. Cela permet d'avoir un modèle du procédé qui est fixe et qui simule la dynamique dans l'ensemble du domaine d'opération pendant que le contrôleur neuronal adaptatif est adapté en permanence dans le domaine d'intérêt et peut suivre les procédés ayant des caractéristiques qui dérivent dans le temps. Le modèle du procédé peut être remis à jour séparément lorsque la dynamique a évolué et que l'on dispose d'une bonne base de données pour en faire l'adaptation.

L'emploi d'un modèle dynamique du procédé dans une structure de contrôle par apprentissage spécialisé permet d'implanter une structure prédictive à un pas en calculant une erreur qui est fonction de la valeur estimée pour la prochaine itération et de la consigne qui couvre cette période. Le modèle dynamique permet d'évaluer une direction de descente qui minimise les poids du contrôleur neuronal et qui pourra être utilisée en conjonction avec un algorithme de recherche unidirectionnelle en estimant l'impact d'une variation de la commande sur la valeur estimée produite par le modèle. Pour prendre en compte les différences entre le modèle et le procédé et minimiser les écarts par rapport à la consigne, la fonction erreur est corrigée par l'écart entre l'état actuel du procédé et l'estimation du modèle pour la même période de temps. La minimisation de cette fonction ne garantit pas qu'il n'y aura aucun écart par rapport à la consigne,

mais, en raison des propriétés des modèles dynamiques par réseaux de neurones, cet écart devrait toujours être très faible.

Le contrôleur neuronal adaptatif impose une limite sur la vélocité de la commande entre deux temps d'échantillonnage. Cette limite est en fait le seul paramètre d'ajustement du contrôleur. La valeur de cette limite doit être déterminée par l'utilisateur et doit refléter la confiance dans le modèle dynamique du procédé ainsi que le degré de robustesse désiré dans le système de contrôle.

Ce travail a mis au point un nouvel algorithme d'entraînement pour réseaux de neurones qui répond aux besoins du contrôleur neuronal adaptatif. Cette méthode se doit d'être performante, robuste, simple à implanter et ne nécessiter aucun ajustement de paramètres afin de procurer de la fiabilité au contrôleur.

Ce nouvel algorithme découle de modifications à l'approche classique des méthodes quasi-Newton, qui ont pour objectif d'accélérer le temps nécessaire à l'entraînement d'un réseau de neurones et de réduire l'espace mémoire requis pour emmagasiner l'information à chaque itération. Ces modifications consistent en de nouvelles approximations de la matrice hessienne qui négligent certaines interactions du deuxième ordre et qui sont construites en prenant en compte la structure du réseau de neurones. En négligeant des interactions du deuxième ordre dans une méthode quasi-Newton, on obtient une direction de descente moins efficace et une augmentation du nombre d'itérations nécessaires pour obtenir une solution. Cet effet est compensé par une réduction du temps de calcul requis pour renouveler l'approximation de la matrice hessienne et par une diminution de l'espace mémoire requis. Les mises à jour des approximations des matrices hessiennes se font par la méthode BFGS qui est reconnue pour être la plus performante des méthodes quasi-Newton.

Les résultats montrent que, lorsque la taille des réseaux de neurones à entraîner est moyenne ou grande, la méthode BFGS-N est plus performante que les méthodes traditionnelles, tout en demandant d'emmagasiner beaucoup moins d'éléments que la méthode BFGS. Pour les

réseaux plus petits, les méthodes du gradient conjugué, BFGS et BFGS-N ont des performances similaires. Le nombre de poids à minimiser à partir duquel la méthode BFGS-N se démarque en terme de performance ne peut être determiné avec exactitude, ce point étant aussi une fonction de la topologie du réseau de neurones. Mais à travers les quatre exemples, la méthode BFGS-N a démontré qu'elle peut être utilisée de façon avantageuse dans toutes les situations. Les deux autres configurations BFGS-L et BFGS-M présentent moins d'intérêt.

Deux exemples d'utilisation du contrôleur neuronal adaptatif ont été développés. Dans le premier exemple, il s'agit de contrôler un bioréacteur CFSTR ayant une dynamique complexe avec différents états d'équilibre et un point de bifurcation. En plus d'avoir une bonne réponse aux changements de consigne, le contrôleur est capable de maintenir le procédé stable à une consigne près du point de bifurcation même en présence d'une perturbation permanente qui introduit des écarts entre le modèle par réseau de neurones et le procédé. L'adaptation du contrôleur aux nouvelles conditions créées par les perturbations est rapide.

L'autre exemple traite d'un réacteur CSTR non-isotherme avec une réaction réversible du premier ordre. Le contrôleur a encore une fois une bonne réponse lors des changements de consigne. Lors de l'introduction d'une perturbation, la réponse du contrôleur neuronal adaptatif est comparable à celle obtenue par une autre équipe qui utilisait aussi des réseaux de neurones dans des conditions similaires avec une approche par modèle interne.

Pour la poursuite des travaux, deux axes sont à explorer. La première devrait porter sur une phase expérimentale à l'échelle laboratoire de l'algorithme de contrôle. Afin de bien évaluer tout le potentiel de la méthode, on devrait choisir un système hautement non linéaire comme l'équilibre du pH d'une solution. Le deuxième axe à explorer est d'étendre cet algorithme pour les systèmes à plusieurs entrées et plusieurs sorties (MIMO) et d'en vérifier la contrôlabilité.

BIBLIOGRAPHIE

AGRAWAL, P., LEE, C., LIM, H.C., RAMKRISHNA, D. (1982) *Theoretical investigation of dynamic behavior of isothermal continuous strirred tank biological reactors*, Chemical Engineering Science, vol. 37, n° 3, p. 453-462.

ARMIJO, L. (1966) *Minimization of function having Lipschtz-continuous first partial derivatives*, Pacific Journal of Mathematic, vol. 16, p. 1-3.

BECKER, S., Le CUN, Y. (1988) *Improving the convergence of back-propagation learning with second order methods*, Proceeding Connectionist Model summer School, Morgan-Kaufman, San mateo, p. 29-37.

BHAT, N., McAVOY, T.J. (1990) *Use of neural nets for dynamic modeling and control of chemical process systems*, Computer and Chemical Engineering, vol. 14, n° 4/5, p. 573-583.

BISHOP, C. (1992) *Exact Calculation of the Hessien Matrix for the Multilayer Perceptron*, Neural Computation, vol. 4, p. 494-501.

BRENGEL, D.D., SEIDER, W.D (1989) *Multistep Nonlinear Predictive Controller*, Industrial Engineering Chemical Research, vol. 28, n° 12, p. 1812-1822.

CHEN, S., BILLING, S.A. (1989) *Representation of non-linear systems: the NARMAX model*, International Journal of Control, vol. 49, n° 3, p. 1013-1032.

CHEN, S., BILLING, S.A., GRANT, P.M. (1990) *Non-linear system identification using neural networks*, International Journal of Control, vol. 51, n° 6, p. 1191-1214.

CHITRA, S.P (1993) *Use Neural Networks for Problem Solving*, Chemical Engineering Progress, avril, p. 44-52.

CUTLER, C. R., RAMAKER, B. L. (1979) *Dynamic matrix control - a computer control algorithm.* AIChE 86th National meeting, Houston, Texas.

DENIS, J.E. JR., SCHNABEL, R.B. (1983) *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, New Jersey, Prentice Hall, 364 p.

ECONOMOU, C.G., MORARI, M., PALSSON, B.O. (1986) *Internal Model Control. 5. Extension to Nonlinear Systems*, Industrial Engineering Chemical Process Design and Development, vol. 25, n° 2, p. 403-411.

EDGAR, T.F., HIMMELBLAU, D.M. (1988) *Optimization of chemical processes*, McGraw-Hill book compagny, New-York, 652 p.

FAHLMAN, S.E. (1988) *An Empirical Study of Learning Speed in Back-Propagation Networks*, Pittsburgh, internal report: CMU-CS-88-162, Carnegie Mellon University, 17 p.

FLETCHER, R., REEVES, C.M. (1964) *Function minimization by conjugate gradients*, Computer Journal, vol. 7, p. 149-154.

GARCIA, C.E., MORARI, M. (1982) *Internal Model Control. 1. A Unifying Review and Some New Results*, Industrial Engineering Chemical Process Design and Developement, vol. 21, n° 2, p. 308-323.

GARCIA, C.E., PRETT, D.V., MORARI, M. (1989) *Model Predictive Control: Theory and Practice- a Survey*, Automatica, vol. 25, n° 3, p. 335-348.

HECHT-NIELSEN, R. (1989) *Neurocomputing*, Addison-Wesley, New York, 433 p.

HERNANDEZ, E., ARKUN, Y. (1990) *Neural Network Modeling and an Extended DMC Algorithm to Control Nonlinear Systems*, Proceeding of the american control conference, Boston, p. 2454-2459.

HIMMELBLAU, D.M. (1989) *Introducing efficient second order effects into Back Propagation learning*, Proceeding of the International joint conference on neural networks, Washington, DC, p. 631-634.

HOSKIN, J.C., HIMMELBLAU, D.M. (1988) *Artificial Neural Network Model of Knowledge Representation in Chemical Engineering*, Computer and Chemical Engineering, vol. 12, n° 9/10, p. 881-890.

JACOBS, R. (1988) *Increased Rates of Convergence Through Learning Rate Adaptation*, Neural Networks, vol. 1, p. 295-307.

JORDAN, M.I. (mai 1988) *Supervised learning and systems with excess degrees of freedom*, COINS Technical Report 88-27, MIT, 40 p.

KINSELLA, J.A. (1992) *Comparison and evaluation of variants of the conjugate gradient method for efficient learning in feed-forward neural networks with backward error propagation*, Network, vol. 3, p. 27-35.

KOLLIAS, S., ANASTASSIOU, D. (1989) *An Adaptative Least Square Algorithm for the Efficient Training of Artificial Neural Networks*, IEEE Transaction on Circuits and Systems, vol. 36, n° 8, p. 1092-1101.

KRAMER, M.A., LEONARD, J.A. (1990) *Diagnosis using backpropagation neural network - Analysis and criticism*, Computer and Chemical Engineering, vol. 14, n° 12, p. 1323-1338.

LATRILLE, E., CORRIEU, G., THIBAULT, J. (1994) *Neural Network Models for Final Process Time Determination in Fermented Milk Production*, Computer and Chemical Engineering, vol. 18, n° 11/12, p. 1171-1181.

LEONARD, J.A., KRAMER, M.A. (1990) *Improvement of the BackPropagation algorithm for training neural networks*, Computer and Chemical Engineering, vol. 14, n° 3, p. 337-341.

LIPPMANN, R.P. (1989) *Rewiew of neural networks for speech recognition*, Neural Computation, vol. 1, p. 1-38.

MC AVOY, T.J., WANG, N.S., NAIDU, S., BATH, N., GUNTER, J., SIMMONS, M. (1989) *Interpreting biosensor data via backpropagation*, Proceedings of the Joint Conference on Neural Networks, Washington, DC, p. 1227-1233.

MINSKY, M., PAPERT, P. (1969) *Perceptrons: An Introduction to Computational Geometry*, Cambridge, MIT Press.

MOALLEMI, C. (1991) *Classifying Cells for Cancer Diagnosis Using Neural Networks*, IEEE Expert, n° 12, p. 8-12.

MORARI, M., ZAFIRIOU, W. (1989) *Robust Process Control*, Englewood Cliffs, New Jersey, PTR Prentice Hall, 488p.

NAHAS, E.P., HENSON, M.A., SEBORG, D.E. (1992) *Nonlinear internal model control strategy for neural network models*, Computer and Chemical Engineering, vol. 16, n° 12, p. 1039-1057.

NARENDRA, K.S., PARTHASARATHY, K. (1990) *Identification and Control of Dynamical Systems Using Neural Networks*, IEEE Transaction on Neural Networks, vol. 1, n° 1, p. 4-27.

NGUYEN, D.H., WIDROW, B. (1990) *Neural networks for self-learning control systems*, IEEE Control Systems Magazine, n° 4, p. 18-23.

NORMANDIN, A., GRANDJEAN, B.P.A., THIBAULT, J. (1993) *PVT Data Analysis Using Neural Network Model*, Industrial Engineering Chemical Research, vol. 32, n° 5, p. 970-975.

PARKER, D.B. (1985) *Learning-logic*, Cambridge, Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT.

PSALTIS, D., SIDERIS, A., YAMAMURA, A.A (1987) *Neural Controllers*, IEEE first International Conference on Neural Networks, San Diego, p. IV-551-558.

PSALTIS, D., SIDERIS, A., YAMAMURA, A.A (1988) *A Multilayered Neural Network Controller*, IEEE Control Systems Magazine, April, p. 17-21.

PSICHOGIOS, D. C., UNGAR, L.H. (1991) *Direct and Indirect Model Based Control Using Artificial Neural Networks*, Industrial Engineering Chemical Research, vol. 30, n° 12, p. 2564-2573.

RENARD, F. (1995) *Réseaux de neurones couplés à la spectrophotométrie ultraviolette pour le contrôle automatique de la qualité des eaux*, Mémoire de maîtrise, Faculté des Sciences Appliquées, Université de Sherbrooke, 79 p.

RUENGLERTPANYAKUL, W., KONSTANTINOV, K.B., YOSHIDA, T. (1992) *Application of neural networks to variables estimation and stage identification in phenylalanime production*, IFAC Modeling and Control of Biotechnical Processes Conference, Colorado, p. 429-432.

RUMELHART, D.E., HINTON, G.E., WILLIAM, R.J. (1986) *Learning Internal Representation by Error Propagation*, Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Cambridge, Rumelhart and McClelland, MIT Press, Tome 1, chapitre 8, p. 318-362.

SAERENS, M., SOCQUER, A.[1989] *A Neural Controller*, Proceedings of the First IEE International Conference on Artificial Neural Networks, London, IEE Press, p.211-215.

SAINT-DONAT, J., BHAT, N., McAVOY, T.J (1991) *Neural net based model predictive control*, International Journal of Control, vol. 54, n° 6, p. 1453-1468.

SCHIFFMANN, W. H., GERRERS, W. H. [1993] *Adaptive Control of Dynamic Systems by Back Propagation Networks*, Neural Networks, vol. 6, p. 517-524.

SEBORG, D.E. (1994) *Experience with nonlinear control and identification strategies*, Control '94, IEE Conference publication No. 389, 21-24 mars, p. 879-886.

SEBORG, D.E., EDGAR, T.F., MELLICHAMP, D.A. (1989) *Process Dynamics and Control*, New York, John Wiley & Son, 701p.

SU, H.T., McAVOY, J., WERBOS, P. (1992) *Long-Term Prediction of Chemical Process Using Recurrent Neural Networks: A Parallel Training Approach*, Industrial Engineering Chemical Research, vol. 31, n° 5, p. 1338-1352.

THIBAULT, J., GRANDJEAN, B.P.A. (1991) *A neural network methodology for heat transfert data analysis*, International Journal of Heat and Mass Transfert, vol. 34, n° 8, p. 2063-2070.

THIBAULT, J., GRANDJEAN, B.P.A. (1992) *Process Control Using Feedforward Neural Networks*, Journal of Systems Engineering, n° 2, p. 198-212.

THOMAS, O., GALLOT, S. (1990) *Ultraviolet multiwavelength absorptiometry (UVMA) for the examination of natural waters and wastewaters; partI: Genaral considerations*, Fresenius Journal of Analytical Chemistry, vol. 338, p. 234-237.

TILFROD, B. (1994) *An Introduction to Neural Networks*, Quantitative Research, Toronto, Nesbitt Burns compagnie, septembre, 12 p.

TOLLENAERE, T. (1990) *SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties*, Neural Networks, vol. 3, p. 561-573.

UNGAR, L.H. (1990) *A bioreactor benchmark for adaptive network-based process control*, Neural networks for control, , W.T. Miller, III, R.S. Sutton and P.J. Werbos, MIT Press, Cambridge, chapitre 16, p. 387-402.

UNGAR, L.H., POWEL, B.A., KAMENS, S.N. (1990) *Adaptive networks for fault diagnosis and process control*, Computer and Chemical Engineering, vol. 14, n° 4/5, p. 561-572.

VAN OOYEN, A., NIENHUIS, B. (1992) *Improving the Convergence of the Back-Propagation Algorithm*, Neural Networks, vol. 5, p. 465-471.

VENKATASUBRAMANIA, V., CHAN, K. (1989*) A neural network methodology for process fault diagnosis*, AIChE Journal, vol. 35, n° 12, p. 1993-2002.

VENKATASUBRIAMANIA, V., VAIDYANATHAN, R.L, YAMAMOTO, Y. (1990) *Process fault detection and diagnosis using neural networks- I. Steady-state process*, Computer and Chemical Engineering, vol. 14, n° 7, p. 699-712.

WATROUS, R.L. (1987) *Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization*, IEEE first International Conference on Neural Networks, San Diego, p. II-619-627.

WERBOS, P.J. (1974) *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences*, Cambridge, thesis in applied mathematic, Harvard University.

YDSTIE, B.E. (1990) *Forecasting and control using adaptive connectionist networks*, Computer and Chemical Engineering, vol. 14, n° 4/5, p. 583-599.

YOU, Y., NIKOLAOU, M. (1993) *Dynamic Process Modeling with Recurrent Neural Network*, AIChE Journal, vol. 39, n° 10, p. 1654-1667.